UNIVERSITY OF ALBERTA

A Local Mesh Refinement Technique for Multi-Grid Method on Elliptic Equations

by

© **Allen K. Chien**

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

Master of Science

IN

APPLIED MATHEMATICS

DEPARTMENT OF MATHEMATICS

EDMONTON, ALBERTA

FALL, 1990

National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-64969-0

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: **Allen K. Chien**

TITLE OF THESIS: **A Local Mesh Refinement Technique for Multi-Grid Method on Elliptic Equations**

DEGREE FOR WHICH THESIS WAS PRESENTED: **Master of Science**

YEAR THIS DEGREE GRANTED: **1990**

PERMISSION IS HEREBY GRANTED TO THE UNIVERSITY OF ALBERTA LIBRARY TO REPRODUCE SINGLE COPIES OF THIS THESIS AND TO LEND OR SELL SUCH COPIES FOR PRIVATE,SCHOLARLY OR SCIENTIFIC RESEARCH PURPOSES ONLY.

THE AUTHOR RESERVES OTHER PUBLICATION RIGHTS, AND NEITHER THE THESIS NOR EXTENSIVE EXTRACTS FROM IT MAY BE PRINTED OR OTHERWISE REPRODUCED WITHOUT THE AUTHOR'S WRITTEN PERMISSION.

SIGNED

PERMANENT ADDRESS:

Department of Mathematics

University of Alberta

Edmonton, Alberta, Canada T6G 2G1

Date

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

THE UNDERSIGNED CERTIFY THAT THEY HAVE READ, AND RECOMMEND TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH FOR ACCEPTANCE, A THESIS ENTITLED **A Local Mesh Refinement Technique for Multi-Grid Method on Elliptic Equations** SUBMITTED BY **Allen K. Chien** IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF **Master of Science** IN APPLIED MATHEMATICS.

Y.S. Wong
Supervisor

S. Sivaloganathan
Supervisor

B. Joe

Date June 18, 1990

To my grandmother

Abstract

In this thesis we investigate the performance of the multi-grid method in the context of local mesh refinements. The method illustrated here is based on the uniform multi-grid FAS scheme; however, the treatment of boundary conditions on internal boundaries introduced by local mesh refinement has been carefully considered. It allows us not only to obtain any desired refinement pattern but also to solve the discretized problems using a hierarchy of only uniform grids (extending over possibly smaller regions). Preliminary numerical experiments indicate that, apart from savings in storage, there are enormous savings in CPU time in comparison with standard multi-grid (where successive grids uniform refinements over the whole region of interest). The approach adopted here is especially useful for problems with boundary layers and in general for problems where rapid variation occurs in a very small region of the domain of interest while the behavior in the remaining part of the domain is smooth.

Acknowledgement

The author wishes to extend his appreciation to his supervisors, Dr. Y. S. Wong and Dr. S. Sivaloganathan, for all of their constant guidance, critical insight, and immeasurable contributions throughout this study. He also gratefully acknowledges the financial support for this research by the mentioned supervisors as well as the Department of Mathematics.

In addition, the author extends his gratitude to his fellow graduate students for having provided him a stimulating atmosphere to work in. He also would like to thank Mrs. Marion Benedict for her kindly help with TeX.

Finally, very special thanks are due his parents and sister for their understanding, encouragement, and support.

TABLE OF CONTENTS

		Page
1. Introduction		**1**
2. Uniform Multi-grid Method		**4**
2.1	Notations	4
2.2	Smoothing Property of Classical Iteration Methods	5
2.3	Uniform Multi-grid Algorithm	10
2.4	Nonlinear Treatment (FAS Scheme)	16
3. Local Mesh Refinement Technique		**21**
3.1	Introduction	21
3.2	Composite Grid Structure	23
3.3	Local Mesh Refinement Algorithm (LMRA)	25
3.4	Local Boundary Treatments	27
4. Numerical Results		**33**
Bibliography		**37**
Appendix: List of a FORTRAN77 Program for LMRA		**40**

1. Introduction.

Consider the elliptic differential equation:

$$\begin{cases} L_\Omega u = f_\Omega, & \text{in } \Omega \\ L_\Gamma u = f_\Gamma, & \text{on } \Gamma \end{cases} \tag{1.1}$$

where,

Ω is a d-dimensional region;

Γ is the boundary of Ω;

$u = u(\mathbf{x}), f_\Omega = f_\Omega(\mathbf{x}), f_\Gamma = f_\Gamma(\mathbf{x})$ are defined on $\mathbf{x} \in \Omega \cup \Gamma$; and

L_Ω, L_Γ are linear or nonlinear operators.

System (1.1) could be discretized somehow, say by the finite difference or finite element method, to get a discretized system:

$$L_h u_h = f_h, \qquad \mathbf{x} \in \Omega_h \subset \Omega \tag{1.2}$$

where $u_h, f_h \in G(\Omega_h)$: the space of gridfunctions defined on a grid Ω_h, and

$$L_h : G(\Omega_h) \to G(\Omega_h)$$

We assume here that the boundary conditions have been discretized and substituted into neighboring interior equations in the usual way.

As the model problem, let's consider the following two-dimensional Poisson equation with Dirichlet boundary conditions:

$$\begin{cases} \Delta u = f, & \text{in } \Omega = (0,1) \times (0,1) \\ u = g, & \text{on the boundary of } \Omega \end{cases} \tag{1.3}$$

Comparing (1.3) with (1.1), we have

$$L_\Omega = \Delta, \quad L_\Gamma = I, \quad \Omega = (0,1) \times (0,1)$$

Discretize the model problem (1.3) using a five-point difference stencil on the uniform grid Ω_h:

$$\Omega_h = \{(x_h, y_h) \mid (x_h, y_h) = (ih, jh), 0 \le i, j \le n\}$$

where the mesh size $h = \frac{1}{n}$, we have the corresponding discretized system (1.2) with

$$L_h = \frac{1}{h^2} \begin{pmatrix} A & I & & & \\ I & A & I & & \\ & \ddots & \ddots & \ddots & \\ & & I & A & I \\ & & & I & A \end{pmatrix}_{(n-1)^2 \times (n-1)^2} \tag{1.4}$$

and

$$A = \begin{pmatrix} -4 & 1 & & & \\ 1 & -4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -4 & 1 \\ & & & 1 & -4 \end{pmatrix}_{(n-1)\times(n-1)} \tag{1.5}$$

Usually, due to the sparseness of the matrix L_h, iteration methods are preferable choices for solving the discretized system (1.2).

In this thesis we examine the multi-grid method, with a view to solving problems where the constraints of the computing facilities (costs and storage) render the use of a uniformly refined hierarchy of grids infeasible. The problems under consideration require a high resolution in very small regions of the domain, and in this context the most efficient approach may be to use a grid hierarchy where successive finer grids extend over possibly smaller regions of the preceding coarser grids. Brandt[6],[7] and McCormick[8] have used this approach. Their methods differ in the type of boundary conditions imposed on the artificial internal boundaries introduced by local mesh refinement. Our approach is a hybrid of these two approaches, i.e. we have adopted not only Brandt's intuitive implementation of the algorithm but also McCormick's careful internal boundary treatments. The reason for doing this is that we try to seek a

straight-forward implementation for the problem solving whilst at the same time trying to avoid introducing false physics into the solution.

Briefly, the contents of the thesis are as follows: in section 2, the smoothing property of classical iteration methods and uniform multi-grid method are discussed; then in section 3, the local mesh refinement technique is elaborated; next, some numerical results derived from using the technique are given in section 4; and finally, a program list (in *FORTRAN77*) for the local mesh refinement algorithm is attached in the Appendix.

2. Uniform Multi-grid Method.

For the time being, to facilitate the discussion, we assume that L_h in (1.2) is a linear operator, the nonlinear treatment will be presented in subsection 2.4.

2.1 Notations.

Let u_h^* be the exact solution of system (1.2), and v_h an approximation of u_h^*. There are two important measures of v_h as an approximation to u_h^*. The first one is the (algebraic) error e_h which is defined by:

$$e_h := u_h^* - v_h \tag{2.1}$$

Therefore, $e_h \in G(\Omega_h)$. Unfortunately, because u_h^* is unknown, the error e_h is just as inaccessible as the exact solution u_h^* itself. However, as an alternative measure, the residual given by:

$$r_h := f_h - L_h v_h \tag{2.2}$$

will give us a computable measure of how well v_h approximates u_h^*.

From (2.1), (2.2), and due to the uniqueness of the solution u_h^*, it is clear that the residual $r_h = 0$ if and only if the error $e_h = 0$. Nevertheless, it is not true that there is a direct connection between the error and residual, i.e. the error e_h may not be small when the residual r_h is small.

Now, by subtracting equation

$$L_h v_h = f_h - r_h$$

from

$$L_h u_h^* = f_h$$

we have the following residual equation:

$$L_h e_h = r_h \tag{2.3}$$

which provides us with the very important fact that the error e_h satisfies the same set of equations as the unknown u_h, when f_h is replaced by the residual r_h. It should be emphasized that the residual equation (2.3) plays a vital role in multi-grid methods which will be discussed shortly.

By solving the residual equation (2.3) for e_h, we may improve the approximation v_h by a new approximation $\bar{v}_h$ defined by:

$$\bar{v}_h := v_h + e_h \tag{2.4}$$

This is just the residual correction idea used by the classical iteration methods. Now, in order to understand the motivation of the multi-grid method, we examine the smoothing property of classical iteration methods.

2.2 Smoothing property of classical iteration methods.

Among the classical iteration methods, we focus on the damped Jacobi iteration method because it allows the most transparent analysis.

Let D_h denote the diagonal part of L_h, i.e.

$$D_h := diag(L_h)$$

The damped Jacobi iteration method for (1.2) is:

$$v_h^{m+1} := v_h^m + \omega D_h^{-1}(f_h - L_h v_h^m) \tag{2.5}$$

where, v_h^m is the mth approximation of v_h to u_h^* on grid Ω_h, and $\omega \in [0,1]$ is a damping parameter.

For exact solution of (1.2), we have:

$$u_h^* := u_h^* + \omega D_h^{-1}(f_h - L_h u_h^*) \tag{2.6}$$

By subtracting (2.5) from (2.6), we get:

$$e_h^{m+1} := (I - \omega D_h^{-1} L_h) e_h^m \tag{2.7}$$

where,

$$e_h^m := u_h^* - v_h^m$$

Denote the iteration matrix as S_h, i.e.

$$S_h := I - \omega D_h^{-1} L_h \tag{2.8}$$

we have:

$$\begin{aligned} e_h^{m+1} &= S_h e_h^m \\ &= S_h^2 e_h^{m-1} \\ &\cdots \\ &= S_h^{m+1} e_h^0 \end{aligned} \tag{2.9}$$

Therefore,

$$\| e_h^m \| \leq \| S_h \|^m \cdot \| e_h^0 \|$$

Practically speaking, the most commonly used vector (or matrix) norm $\| \cdot \|_\infty$ or $\| \cdot \|_2$ may be used here as measure for e_h (or S_h). It follows that if $\| S_h \| < 1$, then the error approaches zero as the iteration proceeds.

Recall the definition of spectral radius

$$\rho(S_h) := \max_i \mid \lambda_i(S_h) \mid \tag{2.10}$$

where $\lambda_i(S_h)$ is the ith eigenvalue of S_h, the following is true:

$$\lim_{m\to\infty} S_h^m = 0 \quad \text{if and only if} \quad \rho(S_h) < 1$$

Therefore, it follows that the error e_h^m converges to zero if and only if the spectral radius of S_h is less than one.

The term $\rho(S_h)$ is also called the convergence factor and could be interpreted as the worst factor by which the error is reduced by each relaxation iteration. Moreover, suppose that $\rho(S_h) < 1$ and let M be the smallest integer satisfying

$$[\rho(S_h)]^M \leq 10^{-t}$$

by solving for M, we have:

$$M \geq \frac{t}{-log_{10}[\rho(S_h)]} \tag{2.11}$$

Therefore, the convergence factor $\rho(S_h)$ could be used to calculate the number of iterations needed to reduce the error by a factor of 10^{-t}. The term $-log_{10}\ [\rho(S_h)]$ on the denominator in (2.11) is called the covergence rate which gives the number of iterations required to reduce the error by one decimal digit. It is obvious from (2.11) that the smaller the value of $\rho(S_h)$, the higher the convergence rate of iterations.

Now, let's take a look at the convergence behavior of our model problem in more detail.

The eigenvalues and eigenfunctions of L_h defined by (1.4) are

$$\lambda_{pq}(L_h) = -\frac{4}{h^2}\left[sin^2(\frac{p\pi h}{2}) + sin^2(\frac{q\pi h}{2})\right] \tag{2.12}$$

$$w_{pq}(L_h) = sin(p\pi x_h)sin(q\pi y_h), \quad (x_h, y_h) \in \Omega_h \tag{2.13}$$

for $1 \leq p, q \leq n-1$.

By (2.8), we have that S_h has the same eigenfunctions as L_h, i.e.

$$w_{pq}(S_h) = w_{pq}(L_h) = sin(p\pi x_h)sin(q\pi y_h), \quad (x_h, y_h) \in \Omega_h$$

and the corresponding eigenvalues of S_h are:

$$\lambda_{pq}(S_h) = 1 - \omega\left[sin^2(\frac{p\pi h}{2}) + sin^2(\frac{q\pi h}{2})\right] \tag{2.14}$$

for $1 \leq p, q \leq n-1$.

It is clear that for convergence of damped Jacobi iteration, it is necessary to have $0 < \omega \leq 1$.

By using the eigenfunction expansion for e_h^0 along $\{w_{pq}(L_h)\}$, we have:

$$e_h^0 := \sum_{1 \leq p,q \leq n-1} \alpha_{pq} w_{pq}(L_h) \tag{2.15}$$

where α_{pq} are the proper weighting factors.

On the right hand side of (2.15), the components $\alpha_{pq} w_{pq}(L_h)$ corresponding to $1 \leq p, q \leq \frac{n}{2} - 1$ are called low frequency or smooth error components of e_h^0 while those corresponding to $\frac{n}{2} \leq p, q \leq n-1$ are called high frequency or oscillatory error components.[6],[7]

Note that

$$w_{pq}(S_h) = w_{pq}(L_h)$$

and

$$S_h w_{pq}(S_h) = \lambda_{pq}(S_h) w_{pq}(S_h)$$

by (2.9), we have:

$$\begin{aligned} e_h^m &= S_h^m e_h^0 \\ &= S_h^m \sum_{1 \leq p,q \leq n-1} \alpha_{pq} w_{pq}(L_h) \end{aligned}$$

$$
\begin{aligned}
&= \sum_{1\le p,q\le n-1} \alpha_{pq} S_h^m w_{pq}(S_h) \\
&= \sum_{1\le p,q\le n-1} \alpha_{pq} \lambda_{pq}^m(S_h) w_{pq}(S_h) \\
&= \sum_{1\le p,q\le n-1} \alpha_{pq} \lambda_{pq}^m(S_h) w_{pq}(L_h) \\
&= \sum_{1\le p,q\le n-1} \lambda_{pq}^m(S_h)\left[\alpha_{pq} w_{pq}(L_h)\right] \qquad (2.16)
\end{aligned}
$$

From (2.16), we may discover some very interesting facts. If we take $p = q = 1$ in (2.14), we have

$$
\begin{aligned}
\lambda_{11}(S_h) &= 1 - \omega\left[sin^2(\frac{\pi h}{2}) + sin^2(\frac{\pi h}{2})\right] \\
&= 1 - 2\omega sin^2(\frac{\pi h}{2}) \\
&\approx 1 - 2\omega(\frac{\pi h}{2})^2 \\
&= 1 - \frac{\omega\pi^2}{2}h^2 \\
&= 1 - O(h^2)
\end{aligned}
$$

which approaches one when $h \to 0$ for any $0 < \omega \le 1$. Therefore, by (2.16), we know that the error component $\alpha_{11} w_{11}(L_h)$ has remained almost undamped after several iterations for any $0 < \omega \le 1$. In other words, no matter how we choose the damping parameter ω, the smooth error components can not be damped effectively. The smaller the mesh size h, the worse the convergence rate for smooth error components.

On the other hand, let's consider the smoothing effects of the oscillatory error components, say, take the extreme case $\lambda_{n-1,n-1}(S_h)$:

$$
\begin{aligned}
\lambda_{n-1,n-1}(S_h) &= 1 - \omega\left[sin^2(\frac{(n-1)\pi h}{2}) + sin^2(\frac{(n-1)\pi h}{2})\right] \\
&= 1 - 2\omega sin^2(\frac{(n-1)\pi h}{2})
\end{aligned}
$$

If we choose $\omega = \frac{1}{2}$. then

$$\begin{aligned}\lambda_{n-1,n-1}(S_h) &= 1 - sin^2(\frac{(n-1)\pi h}{2}) \\ &= cos^2(\frac{(n-1)\pi h}{2}) \\ &= sin^2(\frac{\pi h}{2}) \\ &= O(h^2)\end{aligned}$$

Therefore, by choosing an appropiate ω, the damped Jacobi iteration method is very efficient in reducing high frequency error components.

Hence, we observe that the damped Jacobi-iteration method works very well for the first several iterations to damp the oscillatory error components; however, once the oscillatory errors have been removed, the iteration is much less effective in reducing the remaining smooth error components. We call this property the 'smoothing property' of the damped Jacobi iteration method.

Although we have derived the above smoothing property only for the damped Jacobi iteration method, most of the classical iteration methods have this property, too.[21]

Hence, the motivation is to search for an 'optimal' method with the following properties: first, it should damp the oscillatory error components as effectively as the classical iteration methods; second, with some auxilary treatment, it should eliminate the smooth error components almost as effectively as the oscillatory ones; and finally, for optimality, we want to solve the problem in $O(m)$ work units (where m is the size of the problem), the convergence rate should be independent of mesh size h (or at least asymptotically). We will see that multi-grid methods meet the above criteria.

2.3 Uniform multi-grid algorithm.

The previous subsection showed that most of the classical iteration methods such as damped Jacobi method (with $\omega = \frac{1}{2}$) are quite efficient in smoothing out the oscillatory error components. However, they are much less effective in reducing the smooth error components. This is true especially for the case when the grid size h approaches to zero.

Therefore, in order to damp low frequency errors, we should construct a complementary iteration which can reduce the smooth error components very well. By combining this complementary iteration with certain classical iteration method, we may expect to have powerful algorithm to reduce both the oscillatory and smooth error components effectively.

Such a complementary iteration can be obtained by means of the coarse grid $\Omega_H \subset \Omega_h \subset \Omega$. Generally speaking, we may choose the grid size H such that $H = 2h$, where h is the grid size of its immediate finer grid Ω_h. The reason that we construct such complementary iteration is as follows.

Let $S_h : G(\Omega_h) \to G(\Omega_h)$ be some classical iteration operator, and v_h be an approximation of u_h^*. Then,

$$\bar{v}_h := S_h(v_h, f_h) \tag{2.17}$$

Denote

$$e_h := u_h^* - \bar{v}_h$$

we have

$$L_h e_h = r_h \tag{2.18}$$

where,

$$r_h := f_h - L_h \bar{v}_h \tag{2.19}$$

Hence, if we can solve (2.18) exactly to get e_h^*, we have achieved our aim because

$$u_h^* = \bar{v}_h + e_h^*$$

Unfortunately, to solve (2.18) exactly is as difficult as to solve (1.2). Nevertheless, by the previous discussion, we know that e_h is a smooth function, and so is r_h. Therefore, both of them can be well represented on the coarser grid. It follows that the transferring of the residual to the coarser grid can be done without much loss of the information. Hence, the smooth error components on the fine grid now appear more oscillatory on a coarser grid, and it can be removed more efficiently by classical iterations.

Thus, we have the following *coarse-grid correction algorithm*:

Given approximation v_h to u_h^*,

(1) Calculation of residual:

$$r_h := f_h - L_h v_h;$$

(2) Restriction of residual:

$$r_H := R_h^H r_h;$$

(3) Exact solution of coarse-grid equation:

$$e_H := L_H^{-1} r_H;$$

(4) Prolongation of correction:

$$e_h := I_H^h e_H;$$

(5) Correction of v_h:

$$\bar{v}_h := v_h + e_h$$

where,

$R_h^H : G(\Omega_h) \rightarrow G(\Omega_H)$ is the restriction operator;

$I_H^h : G(\Omega_H) \rightarrow G(\Omega_h)$ is the interpolation operator; and

L_H is the discretized operator of L on grid Ω_H.

Unfortunately, the coarse-grid correction by itself is not convergent since the spectral radius of the coarse grid correction iteration matrix is greater than or equal to 1. However, the two grid method, which is the combination of certain classical iteration method with the coarse-grid method, does give a rapid convergence rate. Moreover, the convergence rate is independent of grid size h (or at least asymptotically).[11]

The algorithm of the *two-grid method* is as follows:

Given approximation v_h to u_h^*,

(1) Pre-relaxation (ν_1 times):

$$\bar{v}_h := S_h^{\nu_1}(v_h, f_h);$$

(2) Coarse-grid correction:

(2.a) $r_h := f_h - L_h\bar{v}_h;$

(2.b) $r_H := R_h^H r_h;$

(2.c) $e_H := L_H^{-1} r_H;$

(2.d) $e_h := I_H^h e_H;$

(2.e) $\hat{v}_h := \bar{v}_h + e_h.$

Due to the fact that step (2.d) may introduce some oscillatory error components back to e_h, we may add another ν_2 relaxations (called post-relaxation) to the algorithm, i.e.

(3) Post-relaxtion (ν_2 times):

$$\tilde{v}_h := S_h^{\nu_2}(\hat{v}_h, f_h).$$

Now, let's look at the two-grid algorithm stated above in more detail. First of all, for steps (1) and (3), due to the smoothing property of most

classical iterations, there is no significant advantage to choose certain method over another among the smoothing methods. Empirically speaking, except for some complicated cases, pointwise or linewise symmetric Gauss-Seidel iterations are the usual choices. Moreover, different iterations may be used for steps (1) and (3), respectively. However, for ease of program implementation, the same smoothing method is frequently used for both steps.

Next, for the restriction operator R_h^H, there are several natural choices. The simplest one is called injection. It is just the case that v_H on the coarse grid simply takes its value directly from the corresponding fine grid point v_h. Therefore, for one-dimensional problems,

$$v_H(iH) := v_h(2ih), \qquad 0 \le i \le \frac{N}{2}$$

and for two-dimensional problems, we have:

$$v_H(iH, jH) := v_h(2ih, 2jh), \qquad 0 \le i, j \le \frac{N}{2} \tag{2.20}$$

An alternative choice is called full-weighting. That is, for one-dimensional problems, it is defined as:

$$v_H(jH) = \frac{1}{4}(v_h((2j-1)h) + 2v_h(2jh) + v_h((2j+1)h)), \qquad 1 \le j \le \frac{N}{2} - 1$$

and for two-dimensional ones, the corresponding relation is:

$$\begin{aligned} v_H(iH, jH) := \frac{1}{16}\big(& v_h((2i-1)h, (2j-1)h) + v_h((2i-1)h, (2j+1)h) \\ & + v_h((2i+1)h, (2j-1)h) + v_h((2i+1)h, (2j+1)h) \\ & + 2(v_h(2ih, (2j-1)h) + v_h(2ih, (2j+1)h) \\ & + v_h((2i-1)h, 2jh) + v_h((2i+1)h, 2jh)) \\ & + 4v_h(2ih, 2jh)\big), \qquad 1 \le i, j \le \frac{N}{2} - 1 \end{aligned} \tag{2.21}$$

It should be pointed out that unless the coefficients of the problem vary wildly, injection is the natural and convenient choice for restriction.

Thirdly, for prolongation operator I_H^h in step (2.d), in order to have the full efficiency, the order of I_H^h should be no less than the order of the differential equation. Nevertheless, practically speaking, higher prolongation order has no obvious advantage. Therefore, for second order problems, corresponding to one- or two-dimensional problems, the linear or bi-linear interpolation is the common choice, i.e. for one-dimensional problems, the prolongation is:

$$\begin{cases} v_h(2jh) := v_H(jH), & 0 \le j \le \frac{N}{2} \\ v_h((2j+1)h) := \frac{1}{2}\big(v_H(jH) + v_H((j+1)H\big), & 0 \le j \le \frac{N}{2} - 1 \end{cases}$$

and for two-dimensional problems, we have

$$\begin{cases} v_h(2ih, 2jh) := v_H(iH, jH), & 0 \le i, j \le \frac{N}{2} \\ v_h((2i+1)h, 2jh) := \frac{1}{2}\big(v_H(iH, jH) + v_H((i+1)H, jH)\big), \\ v_h(2ih, (2j+1)h) := \frac{1}{2}\big(v_H(iH, jH) + v_H(iH, (j+1)H)\big), \\ v_h((2i+1)h, (2j+1)h) := \frac{1}{4}\big(v_H(iH, jH) + v_H((i+1)H, jH) \\ \qquad + v_H(iH, (j+1)H) + v_H((i+1)H, (j+1)H)\big), \\ \qquad 0 \le i, j \le \frac{N}{2} - 1 \end{cases} \tag{2.22}$$

In step (2.c), the operator L_H is the Ω_H approximation to L_Ω and L_Γ. It could be obtained by a similar discretization as L_h on the coarser grid Ω_H. Furthermore, by looking at step (2.c) more carefully, we can immediately discover that, although it needs much less work than doing that on the finer level Ω_h, it is still very expensive to solve the equation $L_H e_H = r_H$ exactly if the grid Ω_H is not coarse enough. However, the idea of the two-grid method provides us with a very natural way to overcome the difficulty here. We may solve $L_H e_H = r_H$ by using the two-grid method on an even coarser grid Ω_{2H}. This

idea can be carried out recursively until we reach the coarsest grid Ω_{H_0}. Then, on that coarsest grid, it should be quite cheap to solve for e_{H_0} exactly.

Finally, there is one important remaining problem to be solved, that is how to get a "good" initial approximation v_h on the grid Ω_h. A good initial approximation should, at least in its early stages, improve the relaxation scheme. A well-known technique to obtain such a good initial approximation is to apply some preliminary iteration on a coarser grid and then interpolate the resulting approximation to the original fine grid. Relaxation on a coarser grid is less expensive since there are fewer unknowns to be updated. Moreover, since the convergence factor behaves like $1 - O(H^2)$ on the coarser grid Ω_H, the coarser grid will have a fairly improved convergence rate.

We have mentioned before that the convergence rate of the two-grid method is independent of mesh spacing. Now, because a multi-grid method is one in which the two-grid method is applied recursively over a sequence of grids, we may expect that it attains grid independent convergence rates, too; and indeed, that is the case.[11]

Figure 2.1 summarizes in flow diagram form the uniform multi-grid algorithm for linear problems.

2.4 Nonlinear treatment and FAS algorithm.

The previously described multi-grid method is only valid for the case when L is linear. However, many of the problems we wish to solve, for example the Navier-Stokes equations, are nonlinear. Therefore, we have to extend the multi-grid approach to treat nonlinear problems.

One approach to deal with nonlinear problems is to use a Newton multi-grid iteration method.[10],[11],[15] Nevertheless, instead of solving nonlinear problems directly, we have to spend considerable effort in carrying out the Newton

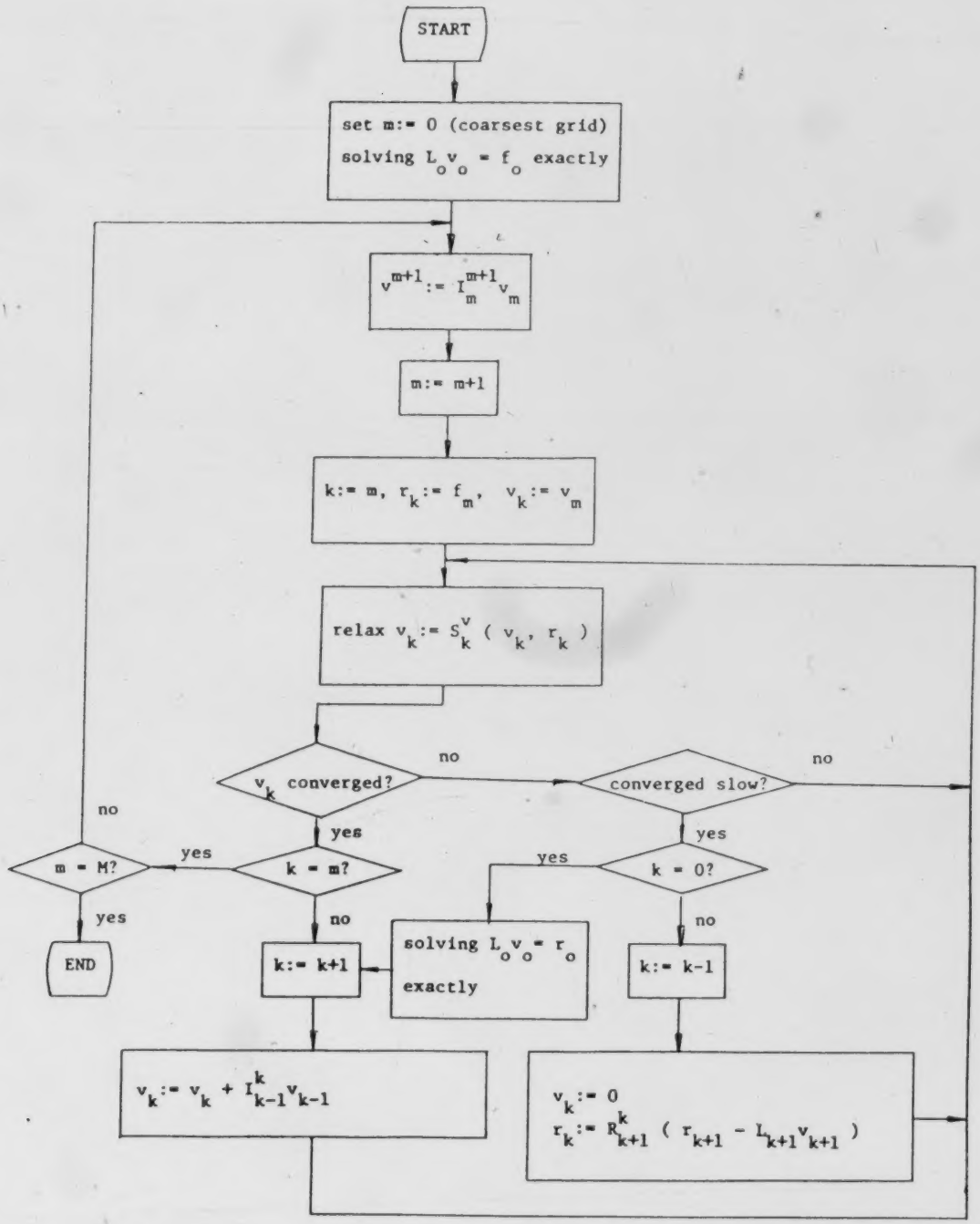


Figure 2.1

Uniform multi-grid algorithm for linear problems

linearization.The direct approach is to use the Full Approximation Scheme (or FAS). It was outlined by Brandt.[6],[7] For our later use, we present the scheme here.

For convenience, we rewrite (1.2) here:

$$L_h u_h = f_h, \qquad \mathbf{x} \in \Omega_h \subset \Omega \tag{2.23}$$

where, u_h, $f_h \in G(\Omega_h)$, and

$$L_h : G(\Omega_h) \to G(\Omega_h)$$

is a nonlinear operator.

The residual equation is as follows:

$$\tilde{L}_h e_h = r_h \tag{2.24}$$

where,

$$\tilde{L}_h e_h \equiv L_h(v_h + e_h) - L_h v_h \tag{2.25}$$

and

$$r_h \equiv f_h - L_h v_h.$$

By (2.25), we know that $\tilde{L}_h = L_h$ if L_h is linear.

From the previous discussion, we know that the error e_h and the residual r_h are smooth functions. Therefore, similar to the linear situation before, we would like to approximate them on the coarser grid Ω_H, where $\Omega_H \subset \Omega_h \subset \Omega$. The approximation to (2.24) on Ω_H is:

$$L_H(R_h^H v_h + e_H) - L_H(R_h^H v_h) = R_h^H r_h \tag{2.26}$$

where L_H is the approximation to L on the coarse grid Ω_H.

Let

$$v_H := R_h^H v_h + e_H$$

$$f_h^H := L_H(R_h^H v_h) + R_h^H r_h$$

by (2.26), we have

$$L_H v_H = f_h^H \tag{2.27}$$

Therefore, rather than solve the error e_H, we compute v_H, the full solution on Ω_H. Once we get the approximation v_H, we prolong the error and update v_h

$$v_h := v_h + I_H^h(v_H - R_h^H v_h) \tag{2.28}$$

which we may expect to be a better approximation v_h to u_h^*.

Now, let's go back to look at (2.27) more carefully. In fact, the right hand side of (2.27) f_h^H can be expressed as

$$f_h^H := f_H + \tau_h^H \tag{2.29}$$

where,

$$\tau_h^H := L_H(R_h^H v_h) - R_h^H(L_h v_h) \tag{2.30}$$

By (2.30), we know that term τ_h^H actually is the truncation error on Ω_H relative to Ω_h. Furthermore, if we consider (2.29) as such a treatment that brings more residual information on fine grid Ω_h down to the coarser grid Ω_H, we may expect to have an approximation on Ω_H with the accuracy of the fine grid Ω_h. It is very important to mention that this idea is the key point of our local mesh refinement technique.

figure 2.2 illustrates the FAS scheme coupled with the full multi-grid approach (FMG).

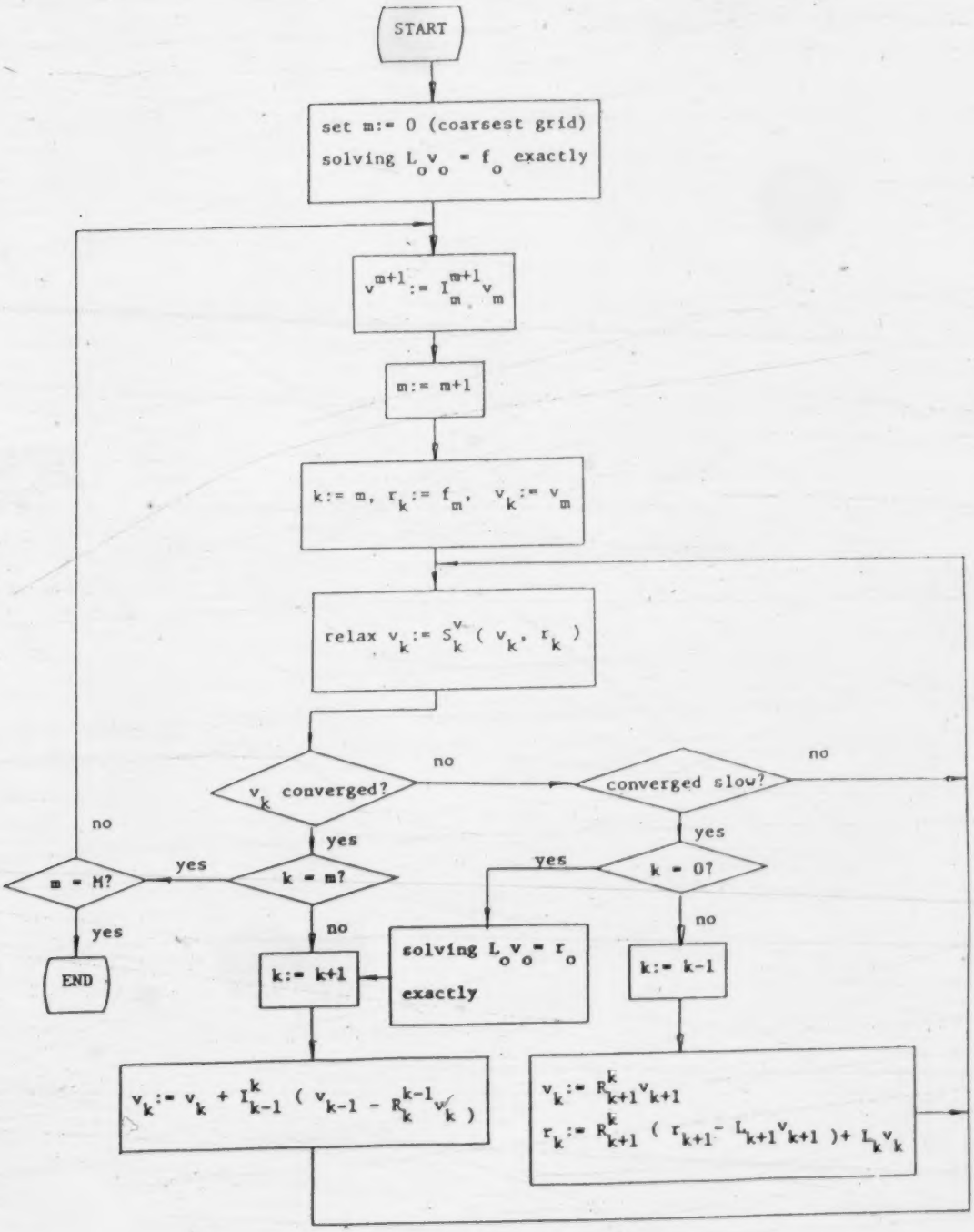


Figure 2.2

Uniform multi-grid FAS algorithm

3. Local Mesh Refinement Technique.

3.1 Introduction.

For many problems, because of the properties of the problems themselves (say, with the unbounded domain, containing singular points such as sources or sinks, or with discontinuous solutions or shock phenomena), or because of the constraints of our computing facilities (memory, project funds, etc.), different resolution in different parts of the domain should be considered.

For example, let's consider our model problem (1.3)

$$\begin{cases} \Delta u = f, & \text{in} \quad \Omega = (0,1) \times (0,1) \\ u = g, & \text{on the boundary of} \quad \Omega \end{cases} \tag{3.1}$$

with

$$f = \left[4\alpha^2\left((1-x)^2 + (1-y)^2\right) - 4\alpha\right] \cdot exp\left[-\alpha\left((1-x)^2 + (1-y)^2\right)\right]$$

and

$$g = \begin{cases} exp\left[-\alpha\left(1 + (1-y)^2\right)\right], & \text{if} \quad x = 0 \quad \text{and} \quad 0 \le y \le 1; \\ exp\left[-\alpha(1-y)^2\right], & \text{if} \quad x = 1 \quad \text{and} \quad 0 \le y \le 1; \\ exp\left[-\alpha\left(1 + (1-x)^2\right)\right], & \text{if} \quad 0 \le x \le 1 \quad \text{and} \quad y = 0; \\ exp\left[-\alpha(1-x)^2\right], & \text{if} \quad 0 \le x \le 1 \quad \text{and} \quad y = 1. \end{cases}$$

where α is a parameter.

We know that the analytical solution of (3.1) is:

$$u(x,y) = exp\left[-\alpha\left((1-x)^2 + (1-y)^2\right)\right] \tag{3.2}$$

However, when α is large, say $\alpha = 100$, the solution $u(x,y)$ will change dramatically near the point (1,1) while it will remain pretty smooth elsewhere (almost zero). Then, it is very important to look closely at the solutions near point (1,1). Thus, the region near (1,1) should be refined.

When we consider areas for local mesh refinement, there are several factors that should be taken into consideration.[8] First, introducing such a local mesh

refinement should not increase the computational cost dramatically. Second, it should not entail reconsideration of the problem entirely when we introduce the local mesh refinement. Next, introducing the local mesh refinement should allow for effective self-adaptive treatments. Finally, we should avoid introducing any false physics into the solutions when we use the local mesh refinement technique.

How about choosing an arbitrary general grid? Practically speaking, although it might be quite flexible, it is very complicated to discretize continuous problems on such a grid, let alone to solve them. Moreover, due to the complicated and tedious bookkeeping work for the grid information, a large amount of memory has to be supplied and much CPU time has to be spent on the extra data management work. The finite element method is a particular example of such a grid structure. Thus, it doesn't seem worthwhile to incorporate such a general grid structure to handle local mesh refinement.

In fact, it is not necessary to use a general grid. The following local mesh refinement technique is flexible enough to permit any desired refinement pattern and to obtain the solution of the discretized problem by applying the multi-grid method on uniform grids only.

Recall that for the uniform multi-grid method, the convergence rate is independent of grid size h and the accuracy is improved with grid refinement. Therefore, a natural idea arises: in order to get an accurate picture of the global solution structure, we refine some regions locally to obtain higher accuracy in those regions. Is there a possible modification to the uniform multi-grid method so that it could meet the following requirements:

(1) to achieve the desired accuracy throughout the entire domain;

(2) to solve the problems on uniform grids only; and

(3) to attain the mesh independent convergence rate (even though the different mesh sizes may appear in the different regions of domain)?

The method introduced here uses the multi-grid method on the so-called "composite-grid", i.e. the union of a series of uniform grids, and is basically similar to Brandt's MLAT algorithm;[2],[7] however, in order to avoid introducing any false physics into the solutions, the flux balance method has been introduced to treat the residuals on the local boundaries. As we will see, by using such local boundary treatment, the whole nonuniform multi-grid iteration will be conservative. This enhances the convergence of the multi-grid method, and provides a more efficient approach than standard multi-grid.

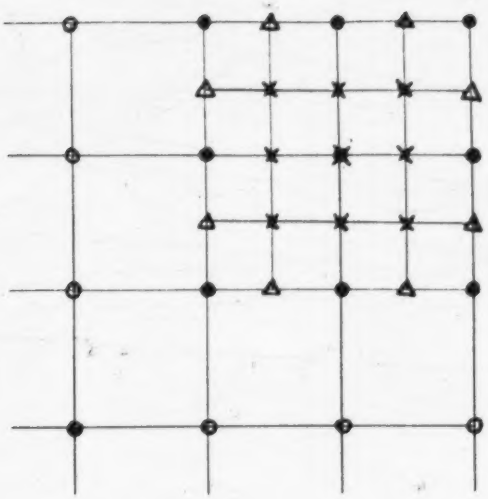
3.2 Composite grid structure.

Now, let's look at the structure of our composite grid. For simplicity, we assume that in the domain of our problem, there is only one local region that should be refined. We organize the composite grid as the combination of a series of uniform grids, say, $\Omega_0, \Omega_1, \ldots, \Omega_M$, where Ω_0 is the coarsest grid and Ω_M is the finest grid. As usual, we choose grid size $h_k = 2h_{k+1}$; moreover, let every other grid line of Ω_{k+1} be the grid line of Ω_k. However, unlike the situation in the uniform multigrid method, grid Ω_{k+1} here may only cover part of Ω_k region, i.e. the region that needs to be refined.

Taking our model problem (3.1) as an example, we may construct our composite grid near (1,1) as shown in *Figure 3.1* (for clear illustration, only two grids Ω_k and Ω_{k+1} have been shown here). We have:

$$\Omega_{com} = \Omega_0 \cup \Omega_1 \cup \ldots \cup \Omega_k \cup \Omega_{k+1} \cup \ldots \cup \Omega_M \tag{3.3}$$

where, Ω_0 is the coarsest grid and Ω_M is the finest grid.



$$\begin{cases} \Omega_k = o \cup \bullet \cup \blacksquare \\ \Omega_{k+1} = \bullet \cup \Delta \cup \blacksquare \cup \times \end{cases}$$

Figure 3.1

Composite grid structure

It is very important to emphasize the role that Ω_k plays. In the regions that the local mesh refinement has to be considered, i.e. if there is a finer grid Ω_{k+1}, Ω_k can be treated as a coarse grid for the correction purpose. On the other hand, in those regions where there is no further resolution needed, Ω_k can be treated as the finest grid there. For example, in *figure 3.1*, those points with $\bullet$ or $\blacksquare$ on grid Ω_k can be considered as the coarse grid points of grid Ω_{k+1}, and those points marked with o can be treated as the finest grid points on grid Ω_k. The first point of view here is especially useful in our local mesh refinement technique.

Note here that in the composite grid Ω_{com}, although Ω_k and Ω_{k+1} share the common geometric points $\bullet$ and $\blacksquare$; generally speaking, the values at those points on different levels Ω_k and Ω_{k+1} are different. On the other hand, just because of those points, we are able to transfer the residuals from local region

grid Ω_{k+1} to coarser grid Ω_k; and also because of those points, we are able to bring the correction back to the local region Ω_{k+1} to achieve higher accuracy. Therefore, they are the key links within the composite grid Ω_{com}. *Figure 3.2* gives us an intuitive picture of this point.

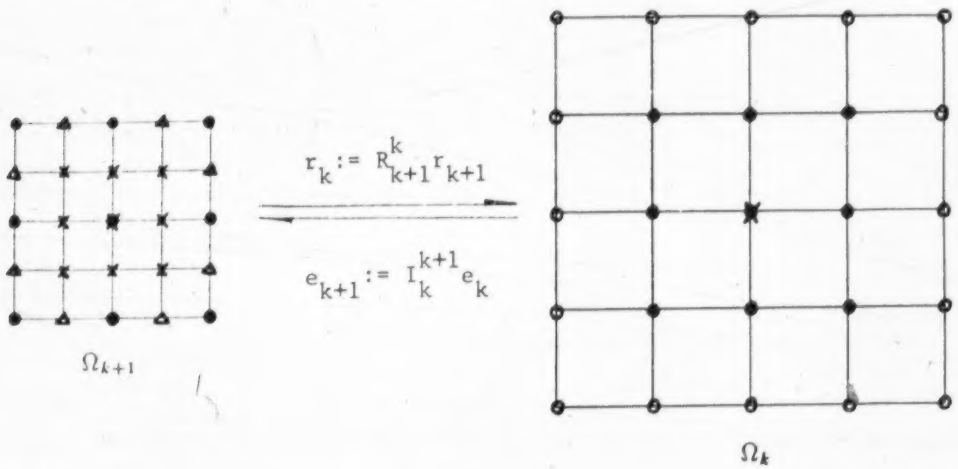


Figure 3.2

3.3 Local mesh refinement algorithm (LMRA).

Recall that the uniform multi-grid FAS algorithm can be summarized as follows:

For a given discretized system

$$L_{k+1} v_{k+1} = f_{k+1}, \qquad \mathbf{x} \in \Omega_{k+1} \subset \Omega \tag{3.4}$$

and an approximation v_{k+1} to u^*_{k+1},

(1) Pre-relaxation (ν_1 times):

$$v_{k+1} := S^{\nu_1}_{k+1}(v_{k+1}, f_{k+1}) \tag{3.5}$$

(2) Coarse-grid correction:

$$v_k := L_k^{-1}\left(L_k(R^k_{k+1} v_{k+1}) + R^k_{k+1}(f_{k+1} - L_{k+1} v_{k+1})\right) \tag{3.6}$$

(3) Correction:

$$v_{k+1} := v_{k+1} + I_k^{k+1}(v_k - R_{k+1}^k v_{k+1}) \tag{3.7}$$

(4) Post-relaxation (ν_2 times):

$$v_{k+1} := S_{k+1}^{\nu_2}(v_{k+1}, f_{k+1}) \tag{3.8}$$

Notice that the right hand side of (3.7) can be express as: $L_k^{-1}(\tau_k^{k+1} + R_{k+1}^k f_{k+1})$, where

$$\tau_k^{k+1} := L_k(R_{k+1}^k v_{k+1}) - R_{k+1}^k(L_{k+1} v_{k+1}) \tag{3.9}$$

It is clear that the term τ_k^{k+1} is the truncation error on Ω_k relative to Ω_{k+1}. Thus, with the help of the composite grid, the idea above results in the following *local mesh refinement algorithm (LMRA)* on the composite grid Ω_{com}:

Given an approximation v_{k+1} to u_{k+1}^* on the local region grid Ω_{k+1},

(1) Pre-relaxation on local region grid Ω_{k+1} (ν_1 times):

$$v_{k+1} := S_{k+1}^{\nu_1}(v_{k+1}, f_{k+1});$$

(2) Coarse-grid correction:

(a) Calculate residuals on the inner points of local region Ω_{k+1} (corresponding to points $\times$ and $\ast$):

$$r_{k+1} := f_{k+1} - L_{k+1} v_{k+1};$$

(b) By using the flux balance method, calculate residuals on those local boundary points corresponding to $\bullet$:

$$r_{k+1} := f_{k+1} - \bar{L}_{k+1}(v_{k+1}, v_k)$$

where $\bar{L}_{k+1} : G(\Omega_{com}) \to G(\Omega_{com})$ is an approximation to L on Ω_{com};

(c) Restrict all residuals obtained by both (a) and (b) to coarse grid Ω_k:

$$r_k := R_{k+1}^k r_{k+1};$$

(d) Solve v_k on coarse grid Ω_k:

$$v_k := L_k^{-1}(L_k(R_{k+1}^k v_{k+1}) + r_k);$$

(3) Correction:

$$v_{k+1} := v_{k+1} + I_k^{k+1}(v_k - R_{k+1}^k v_{k+1});$$

(4) Post-relaxation (ν_2 times):

$$v_{k+1} := S_{k+1}^{\nu_2}(v_{k+1}, f_{k+1}).$$

There are some comments about the algorithm above. First of all, notice that in each step, we solve our problems on uniform grids only, that means that we discretize the continuous problems and make the relaxation sweeps on the uniform grids only. It definitely makes the problem solving much easier than the use of any general grid structures.

Secondly, the algorithm does fit self-adaptive needs. We may detect those local regions which need to have higher resolutions by applying Richardson extrapolation on the error[8] or by some optimization technique on the mesh size and problem approximation order.[2],[6],[7]

Next, the first approximation v_{k+1} to u_{k+1}^* on Ω_{k+1} could be obtained by the linear or bi-linear interpolation from solution v_k on grid Ω_k. As we did before for the uniform multi-grid algorithm, we may get v_k by using the full multi-grid method.

Finally, in the coarse-grid correction step, the calculation of the residuals on local boundaries is the key role of our local mesh refinement technique. Thus, we are going to look at it very carefully in the next subsection.

3.4 Local boundary treatments.

For the residuals on the inner points of local mesh regions, we may use the same difference star for Lu as we do on the coarse grid Ω_k. Let's take our

model problem as an example. We use the finite volume flux balance method to get the five-point finite difference discretization of Δu on the inner points of local mesh region. For any finite volume $S \subset \Omega$, let ∂S be the boundary of S. Then, by Green's Theorem, we have

$$\iint_S \Delta u \, dx \, dy = \int_{\partial S} \frac{\partial u}{\partial n} \, dl \tag{3.10}$$

where $\mathbf{n}$ is the normal direction of the boundary ∂S.

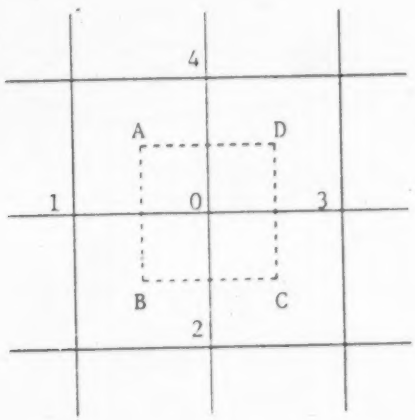


Figure 3.3

Finite volume method for inner points

By *figure 3.3*, for inner points × and ■ of the local region, we choose the finite volume S_0 as the area enclosed by ∂S_0 which is $ABCDA$. Now, by using the following discretization to approximate $\int_{\partial S_0} \frac{\partial u}{\partial n} \, dl$:

$$\begin{aligned}
\int_{\partial S_0} \frac{\partial u}{\partial n} \, dl &= \int_{AB} \frac{\partial u}{\partial n} \, dl + \int_{BC} \frac{\partial u}{\partial n} \, dl + \int_{CD} \frac{\partial u}{\partial n} \, dl + \int_{DA} \frac{\partial u}{\partial n} \, dl \\
&\approx \frac{v_{k+1}^{(1)} - v_{k+1}^{(0)}}{h_{k+1}} \cdot h_{k+1} + \frac{v_{k+1}^{(2)} - v_{k+1}^{(0)}}{h_{k+1}} \cdot h_{k+1} \\
&\quad + \frac{v_{k+1}^{(3)} - v_{k+1}^{(0)}}{h_{k+1}} \cdot h_{k+1} + \frac{v_{k+1}^{(4)} - v_{k+1}^{(0)}}{h_{k+1}} \cdot h_{k+1} \\
&= v_{k+1}^{(1)} + v_{k+1}^{(2)} + v_{k+1}^{(3)} + v_{k+1}^{(4)} - 4v_{k+1}^{(0)}
\end{aligned} \tag{3.11}$$

and from (3.1) and (3.10), we have the following five-point finite difference equation:

$$v_{k+1}^{(1)} + v_{k+1}^{(2)} + v_{k+1}^{(3)} + v_{k+1}^{(4)} - 4v_{k+1}^{(0)} = f_{k+1}^{(0)} \tag{3.12}$$

where,

$$f_{k+1}^{(0)} = \iint_{S_0} f \, dx \, dy.$$

(The superscripts above correspond to the number of geographic positions.)

However, for the local boundary points of the composite grid (but not on the corners), because of the different grid sizes presented in the composite grid Ω_{com}, we can not use the same strategy above to treat the residuals here. Therefore, a special treatment should be applied. Furthermore, in order to avoid introducing any false physics, we have to force the discretized system to be conservative. Thus, we construct the finite volume as illustrated in *Figure 3.4*.

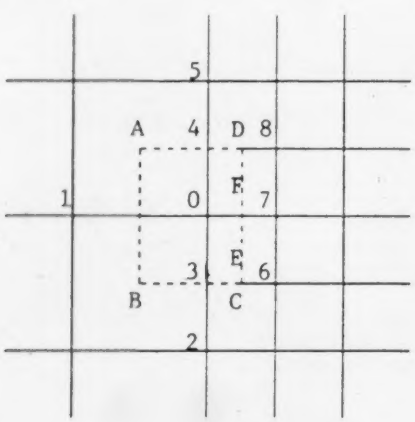


Figure 3.4

Seven-point finite volume method

for inner boundary (but not corner) points

From *Figure 3.4*, we have:

$$\int_{\partial S_0} \frac{\partial u}{\partial n}\, dl = \int_{AB} \frac{\partial u}{\partial n}\, dl + \int_{BC} \frac{\partial u}{\partial n}\, dl + \int_{CD} \frac{\partial u}{\partial n}\, dl + \int_{DA} \frac{\partial u}{\partial n}\, dl$$

Then, we approximate $\int_{AB} \frac{\partial u}{\partial n}\, dl$ by $\frac{v_k^{(1)} - v_{k+1}^{(0)}}{h_k} \cdot h_k$, $\int_{BC} \frac{\partial u}{\partial n}\, dl$ by $\frac{v_{k+1}^{(2)} - v_{k+1}^{(0)}}{2h_{k+1}} \cdot (\frac{h_k}{2} + \frac{h_{k+1}}{2})$, and $\int_{DA} \frac{\partial u}{\partial n}\, dl$ by $\frac{v_{k+1}^{(5)} - v_{k+1}^{(0)}}{2h_{k+1}} \cdot (\frac{h_k}{2} + \frac{h_{k+1}}{2})$. For $\int_{CD} \frac{\partial u}{\partial n}\, dl$, however, we can not simply approximate it by $\frac{v_{k+1}^{(7)} - v_{k+1}^{(0)}}{h_{k+1}} \cdot 2h_{k+1}$ because it will destroy the conservation of the discrete system.[8] Therefore, we modify it by letting

$$\int_{CD} \frac{\partial u}{\partial n}\, dl \approx \frac{v_{k+1}^{(6)} - v_{k+1}^{(3)}}{h_{k+1}} \cdot \frac{1}{2} h_{k+1} + \frac{v_{k+1}^{(7)} - v_{k+1}^{(0)}}{h_{k+1}} \cdot h_{k+1} + \frac{v_{k+1}^{(8)} - v_{k+1}^{(4)}}{h_{k+1}} \cdot \frac{1}{2} h_{k+1} \tag{3.13}$$

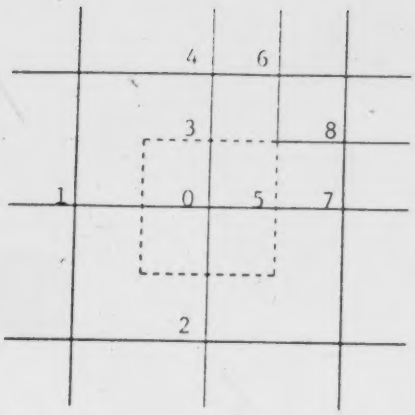
This yields the following approximation $\bar{L}_{k+1}(v_{k+1}, v_k)$ to Lu on the inner boundary points (excluding the corners):

$$\begin{aligned} \bar{L}_{k+1}(v_{k+1}, v_k) \approx\ & (v_{k+1}^{(1)} + v_{k+1}^{(7)} - 2v_{k+1}^{(0)}) + \frac{3}{4}(v_{k+1}^{(2)} + v_{k+1}^{(5)} - 2v_{k+1}^{(0)}) \\ & + \frac{1}{2}(v_{k+1}^{(6)} + v_{k+1}^{(8)} - v_{k+1}^{(3)} - v_{k+1}^{(4)}) \end{aligned} \tag{3.14}$$

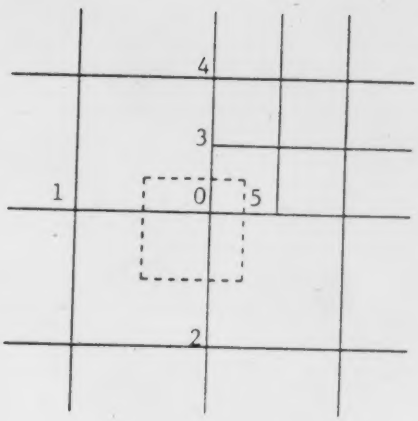
(Here, we have used $h_k = 2h_{k+1}$ for the simplification.)

For the corner points on the inner boundaries of the composite grid, there are several choices. First, similar to the treatment above, we may construct the finite volume as in *Figure 3.5* (a). Then,

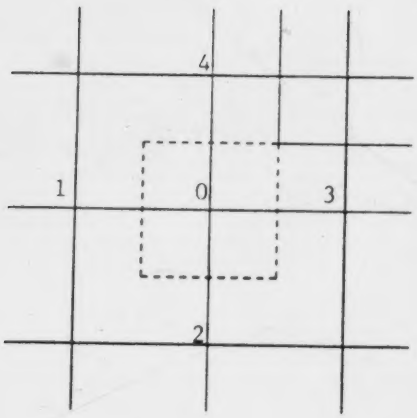
$$\begin{aligned} \bar{L}_{k+1}(v_{k+1}, v_k) \approx\ & \frac{v_k^{(1)} - v_{k+1}^{(0)}}{h_k} \cdot h_k + \frac{v_k^{(2)} - v_{k+1}^{(0)}}{h_k} \cdot h_k \\ & + \frac{v_{k+1}^{(7)} - v_{k+1}^{(0)}}{2h_{k+1}} \cdot (\frac{h_k}{2} + \frac{h_{k+1}}{2}) + \frac{v_{k+1}^{(8)} - v_{k+1}^{(3)}}{2h_{k+1}} \cdot \frac{h_{k+1}}{2} \\ & + \frac{v_{k+1}^{(6)} - v_{k+1}^{(5)}}{2h_{k+1}} \cdot \frac{h_{k+1}}{2} + \frac{v_{k+1}^{(4)} - v_{k+1}^{(0)}}{2h_{k+1}} \cdot (\frac{h_k}{2} + \frac{h_{k+1}}{2}) \\ =\ & (v_k^{(1)} + v_k^{(2)} - 2v_{k+1}^{(0)}) + \frac{3}{4}(v_{k+1}^{(4)} + v_{k+1}^{(7)} - 2v_{k+1}^{(0)}) \\ & + \frac{1}{4}(v_{k+1}^{(6)} + v_{k+1}^{(8)} - v_{k+1}^{(3)} - v_{k+1}^{(5)}) \end{aligned} \tag{3.15}$$



(a)



(b)



(c)

Figure 8.5

Finite volume method

for inner boundary corner points

Another choice, illustrated in *Figure 3.5* (b), is:

$$\begin{aligned}\bar{L}_{k+1}(v_{k+1}, v_k) \approx & \frac{v_k^{(1)} - v_{k+1}^{(0)}}{h_k} \cdot (\frac{h_k}{2} + \frac{h_{k+1}}{2}) + \frac{v_k^{(2)} - v_{k+1}^{(0)}}{h_k} \cdot (\frac{h_k}{2} + \frac{h_{k+1}}{2}) \\ & + \frac{v_{k+1}^{(5)} - v_{k+1}^{(0)}}{h_{k+1}} \cdot (\frac{h_k}{2} + \frac{h_{k+1}}{2}) + \frac{v_{k+1}^{(3)} - v_{k+1}^{(0)}}{h_{k+1}} \cdot (\frac{h_k}{2} + \frac{h_{k+1}}{2}) \\ = & \frac{3}{4}(v_k^{(1)} + v_k^{(2)} - 2v_{k+1}^{(0)}) + \frac{3}{2}(v_{k+1}^{(3)} + v_{k+1}^{(5)} - 2v_{k+1}^{(0)}) \end{aligned} \tag{3.16}$$

The third choice, shown in *Figure 3.5* (c), is:

$$\begin{aligned}\bar{L}_{k+1}(v_{k+1}, v_k) \approx & \frac{v_k^{(1)} - v_{k+1}^{(0)}}{h_k} \cdot h_k + \frac{v_k^{(2)} - v_{k+1}^{(0)}}{h_k} \cdot h_k \\ & + \frac{v_{k+1}^{(3)} - v_{k+1}^{(0)}}{2h_{k+1}} \cdot 2h_{k+1} + \frac{v_{k+1}^{(4)} - v_{k+1}^{(0)}}{2h_{k+1}} \cdot 2h_{k+1} \\ = & v_k^{(1)} + v_k^{(2)} + v_{k+1}^{(3)} + v_{k+1}^{(4)} - 4v_{k+1}^{(0)} \end{aligned} \tag{3.17}$$

Due to the complicated structure of the multigrid method, we have not provided a rigorous convergence proof for our local mesh refinement algorithm; however, as we will see in the next section, the numerical results indicate that the convergence characteristics of the standard multi-grid method are maintained, but with substantial savings in storage and a very significant reduction in CPU time. Nevertheless, due to the structure of the composite grid, the definition of a work unit should be modified to be the computational work of one sweep on the whole composite grid Ω_{com}, not on the finest grid Ω_M alone.

4. Numerical Results.

For the model problem (3.1), as we mentioned before, when α gets quite big, the variation of the solutions near the upper right hand corner will be more rapid. Therefore, the grids near that corner should be refined to a certain degree to get more accurate solutions within that region.

For multi-grid method purposes, we construct the first several grids in the uniform way. For our model problem, we let the first three grids be uniform, and the numbers of grid points are 3×3, 5×5, and 9×9, respectively. Then, we refine the upper right hand corner twice to get another two finer local grids, 9×9 each. The composite grid Ω_{com} is shown in *Figure 4.1*. The coarsest grid shown here is the third grid and the finest one is the fifth grid.

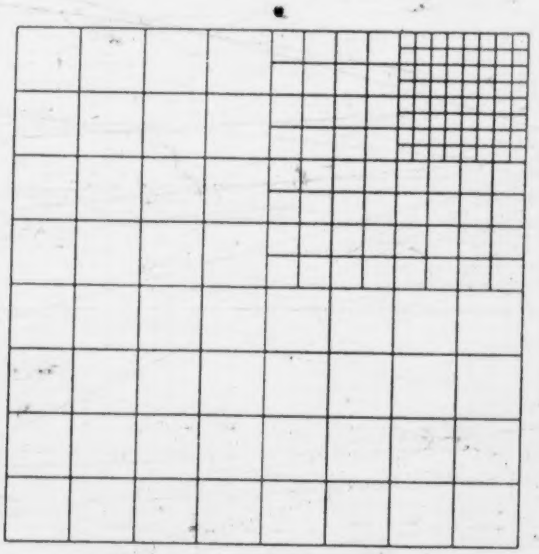


Figure 4.1

A composite grid Ω_{com} for model problem

For $\alpha = 100$, by using injection (2.20) as the restriction R_h^H and bi-linear interpolation (2.22) as the prolongation I_H^h, we have the results shown in *Table 4.1*, and the corresponding graphs of solution are shown in *Figure 4.2*. Moreover, for the convenience of comparison, we give the results derived from applying uniform multi-grid method onto the same problem in *Table 4.2*.

Thus, from *Table 4.1* and *Table 4.2*, it is obvious that our local mesh refinement method is very successful in treating problems where there is rapid variation in certain regions of the domain: the mesh-size independent covergence property has been retained and the higher accuracy has been achieved in the local region as well as the result derived from using the uniform multi-grid method. Moreover, the significant achievement here is the great saving in CPU time. By comparing the time spent on grids 4 and 5 in both tables, we found that the ratio of the time spent on the same finest grid between uniform and composite grids approximately equals the ratio of the areas both grids cover.

In section 3.4, we have discussed the three possible local corner point treatments. The results above have been obtained by using the local boundary treatment choice (b). If we use choice (a) or (c), then we may get the following result of the corresponding absolute errors: $0.435296e-2$ and $0.435372e-2$. The small difference among these three choices is probably due to the small influence of these corner points (if the local region has sufficient grid points).

Finally, we point out that because the coefficients of our model problem are very smooth, numerical results also show that there is no obvious difference between the choice of injection or full-weighting as the restriction operator.

No. of Grid Points	No. of Grid Level	No. of Iterations	CPU (millisec.)	Error $\lvert v_k - u^* \rvert$
3×3	1	2	24	-
5×5	2	3	121	-
9×9	3	4	654	0.706677e-1
9×9	4	4	829	0.160898e-1
9×9	5	4	1017	0.434965e-2

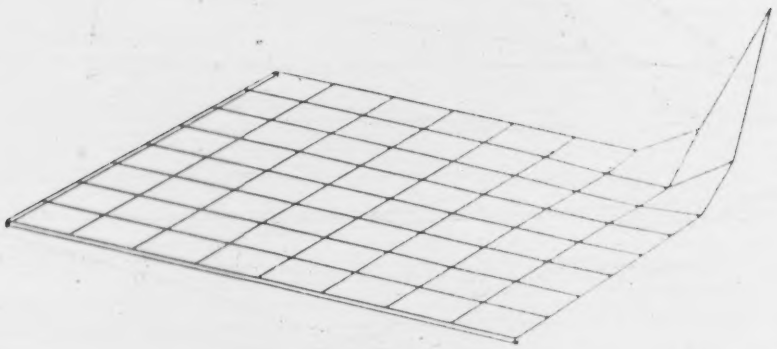
Table 4.1

Local mesh refinement algorithm on model problem ($\alpha = 10^2$)

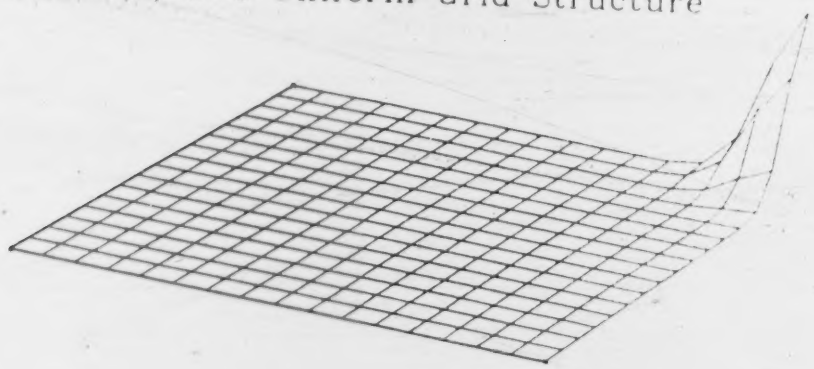
No. of Grid Points	No. of Grid Level	No. of Iterations	CPU (millisec.)	Error $\lvert v_k - u^* \rvert$
3×3	1	2	24	-
5×5	2	3	121	-
9×9	3	4	654	0.706677e-1
17×17	4	4	3485	0.160957e-1
33×33	5	4	15851	0.438037e-2

Table 4.2

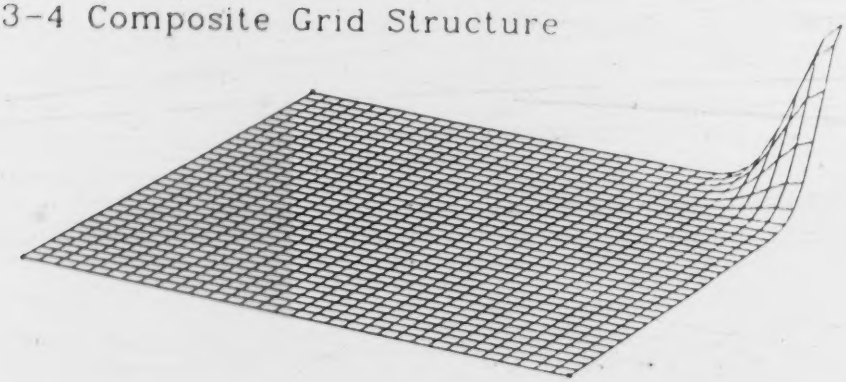
Uniform multi-grid method on model problem ($\alpha = 10^2$)



(a) 3-3 Uniform Grid Structure



(b) 3-4 Composite Grid Structure



(c) 3-5 Composite Grid Structure

Figure 4.2

Bibliography

[1] Auzinger and H. J .Stetter, *Defect correction and multi-grid iterations*, in Multi-grid methods, Proc. conference held at Köln-Porz, November 23-27, 1981, W. Hackbusch and U. Trottenberg, eds., Lecture Notes in Mathematics 960, Springer-Verlag, Berlin, 1982, pp.327-351.

[2] D. Bai and A. Brandt, *Local mesh refinement multi-level techniques*, SIAM J. Sci. Stat. Comput. 8(1987), pp.109-134.

[3] R. E. Bank, A. H. Sherman and A. Weiser, *Refinement algorithms and data structures for regular local mesh refinement*, in Scientific Computing, R. S. Stepleman, ed., North-Holland, Amsterdam, 1983.

[4] D. Braess, *The convergence rate of a multi-grid method with Gauss-Seidel relaxation for the Poisson equation (revised)*, Math. Comp., 42(1984), pp.505-519.

[5] D. Braess, W. Hackbusch and U. trottenberg, eds., *Advances in multi-grid methods*, Proc. conference held in Oberwolfach, December 8-13, 1984, Notes on Numerical Fluid Mechanics, Vol.11, Vieweg, Braunschweig, 1984.

[6] A. Brandt, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31(1977), pp.330-390.

[7] A. Brandt, *Multi-level adaptive techniques (MLAT) for partial differential equations: ideas and software*, in Mathematical Software III, J. R. Rice, ed., Academic Press, New York, 1977, pp.277-318.

[8] R. Ewing, S. F. McCormick and J. Thomas, *The fast adaptive composite grid method for solving differential boundary value problems*, Proc. Fifth ASCE-EMD Speciality Conference, August 1984.

[9] H. Foerster and K. Witsch, *Multi-grid software for the solution of elliptic problems on rectangular domains: MG00 (Release 1)*, in Multi-grid methods, Proc. conference held at Köln-Porz, November 23-27, 1981, W. Hackbusch and U. Trottenberg, eds., Lecture Notes in Mathematics 960, Springer-Verlag, Berlin, 1982.

[10] L. Fuchs, *A Newton-multi-grid method for the solution of non-linear partial differential equations*, in Boundary and Interior Layers — Computational and Asymptotic Methods, J. J. H. Miller, ed., Boole Press, Dublin, 1980, pp.291-296.

[11] W. Hackbusch, *Multi-grid methods and applications*, Springer Series in Comp. Math. 4, Springer-Verlag, Berlin, 1985.

[12] W. Hackbusch, *Local defect correction method and domain decomposition techniques*, in Computing, suppl.5, Defect Correction Methods, K. Böhmer and H. J. Stetter, eds., Springer-Verlag, Wien, 1984, pp.89-113.

[13] W. Hackbusch, *Convergence of multi-grid iterations applied to difference equations*, Math. Comp. 34(1980), pp.425-440.

[14] D. A. H. Jacobs, *Some iteratve methods: past, present, future*, IEEE Colloquium on Numerical Solution Technique for Large Sparse Systems of Equations, London, November 1983.

[15] D. C. Jespersen, *Multi-grid methods for partial differential equations*, in Studies in Numerical Analysis, G. Golub, ed., MAA, 1984.

[16] C. Q. Liu and S. F. McCormick, *Multi-grid elliptic grid generation and the fast adaptive composite grid method for solving transonic potential flow equations*, in Multigrid methods, theory, applications and supercomputing, S. F. McCormick. ed., Marcel Dekker, New York, 1988, pp.365-387.

[17] S. F .McCormick, *Fast adaptive grid (FAC) methods: Theory for the variational case*, in Computing, Suppl.5, Defect Correction Methods, K. Böhmer and H. J. Stetter, eds., Springer-Verlag, Vienna, 1984.

[18] J. R. van Rosendale, *Algorithms and data structures for adaptive multi-grid elliptic solvers*, Appl. Math. Comp., 13, Proc. Internat. Multi-grid Conference, April 6-8, 1983, Copper Mountain, Co., S. F. McCormick and U. Trottenberg, eds , North-Holland, Amsterdam, 1983, pp.453-470.

[19] G. J. Shaw and S. Sivaloganathan, *On the smoothing properties of the simple pressure-correction algorithm*, Internat. J. Numer. Methods Fluids, Vol.8, No.4, 1988, pp.441-461.

[20] S. Sivaloganathan and G. J. Shaw, *A multi-grid method for recirculating flows*, Internat. J. Numer. Methods Fluids, Vol.7, 1987, pp.417-440.

[21] D. M. Young and R. T. Gregory, *A survey of numerical mathematics, Vol.1&2*, Addison-Wesley, 1972.

[22] T. A. Zang, Y. S. Wong and M. Y. Hussaini, *Spectral multi-grid methods for elliptic equations*, J. Comput. Phys., 48(1982), pp.485-501.

[23] T. A. Zang, Y. S. Wong and M. Y. Hussaini, *Spectral multi-grid methods for elliptic equations II*, NASA-CR-172131, ICASE, NASA Langley Research Center, 1983.

Appendix: List of a FORTRAN77 Program for LMRA

Listing of LMRA FOR at 03:15:23 on APR 12, 1990 for CC1d#KCHN on UALTAMTS

```
      1            PROGRAM MUGFAS
      2            IMPLICIT REAL*8 (A-H,O-Z)
      3            COMMON /NOPDES/ MXNOG,NPDES
      4      C
      5      C     ABSTRACT
      6      C     ********
      7      C
      8      C     Functions
      9      C     ---------
     10      C     This program is to solve linear or non-linear 2-D elliptic
     11      C     P D Es (up to two dependent variables).  The FAS schems on
     12      C     composite grid has been coded.  Main sweeping procedure is
     13      C     5-point varied Gauss-Seidel relaxation.  Several treatments
     14      C     for the local interior boundary have been considered.
     15      C     Moreover, the adaptive mesh refinement has been installed
     16      C     for solving shock problems.
     17      C
     18      C     Remarks
     19      C     -------
     20      C     When the real problem has been attacked, the only subroutines
     21      C     which have to be changed to fit the specified needs are the
     22      C     followings:  FN, BCS, FRHS, and DEF, corresponding to the
     23      C     equations, boundary conditions, right hand sides of the
     24      C     equations, and the adjusted R H S  respectively.
     25      C
     26      C
     27      C     PRE-TREATMENT
     28      C     -------------
     29            CALL PRESET
     30      C
     31      C
     32      C     MAIN BODY
     33      C     ---------
     34            DO 100 IGRID=1,MXNOG
     35      C
     36              CALL TIME(0)
     37              CALL MG(IGRID)
     38              CALL TIME(1,0,NTIM)
     39              WRITE (6,9000) IGRID,NTIM
     40      C
     41              IF (IGRID.LT.MXNOG) THEN
     42                CALL PROL(IGRID,0)
     43              ENDIF
     44      C
     45        100 CONTINUE
     46      C
     47      C
     48      C     POST-TREATMENT
     49      C     --------------
     50            CALL POSTIN
     51      C
     52        9000 FORMAT(1X,/,'Time needed to get solution on grid ',I2,
     53           &                ' is  ',I10,' CPU millisec ',1X,/)
     54      C
     55            STOP
     56            END
     57      C
     58      C
     59      C
     60            SUBROUTINE PRESET
     61            IMPLICIT REAL*8 (A-H,O-Z)
     62            COMMON U(50000,2),NX(10),NY(10),IST(10)
     63            COMMON /RANGE/ IB(10),IE(10),JB(10),JE(10)
     64            COMMON /DEFF/ FF(50000,2)
     65            COMMON /SIZE/ MAXTOT,MAXSUB
     66            COMMON /RESTP/ R1,R2,R3
     67            COMMON /PARAM/ EPS
     68            COMMON /UNILEV/ LEV
     69            COMMON /RB/ ID
     70            COMMON /NOPDES/ MXNOG,NPDES
     71            COMMON /MISC/ TOLRES,MAXITS,KREST,NCGC
     72            COMMON /RELP/ KREL,NPRE,NPOST,NCOARS
     73            COMMON /OMEGA/ W(2)
     74            COMMON /CORNER/ NCOR
     75            LOGICAL IDIOT
     76      C
     77      C     EXAMPLE GRID DATA (on uniform grid)
     78      C     ***********************************
     79      C     DATA IB/ 1,1,1,1,1,1,1,1,1,1/
     80      C     DATA IE/ 5,9,17,33,65,129,257,513,1025,2049/
     81      C     DATA JB/ 1,1,1,1,1,1,1,1,1,1/
     82      C     DATA JE/ 5,9,17,33,65,129,257,513,1025,2049/
     83      C
     84      C
     85      C     PARAMETER INPUTS
     86      C     ****************
     87      C     EPS  parameter of P D E
     88            READ(5,*) EPS
     89      C     --------
     90      C     GRID ARRANGEMENT  I -- X-axis, J -- Y-axis
     91            READ(5,*) IB
     92            READ(5,*) IE
     93            READ(5,*) JB
     94            READ(5,*) JE
     95      C     --------
     96      C     LEV  grid level no. above which the nonuniform grids begin
     97            READ(5,*) LEV
     98      C     --------
     99      C     NPDES  no of P D E s in system
    100            READ(5,*) NPDES
    101      C     --------
    102      C     MXNOG  max no of grids allowed in sequence
    103            READ(5,*) MXNOG
    104      C     --------
    105      C     MAXITS  max no of FAS iterations without convergence
    106            READ(5,*) MAXITS
    107      C     --------
    108      C     TOLRES  tolerance for residual norm reduction
    109            READ(5,*) TOLRES
    110      C     --------
    111      C     IDIOT  F - the user has some knowledge of MG
    112      C            T - the user has no knowledge of MG and the
    113      C                values given for the remaining arguments
    114      C                are overwritten to give a robust method
    115            READ(5,*) IDIOT
    116      C     --------
    117      C
    118      C
    119            IF (IDIOT) THEN
    120      C     -----------------------------------------------
    121      C     If the user has no knowledge about the system,
    122      C     Set Robust Parameter Choice
```

```
C     ------------------------------------------------
C       (Set pre-, post-, and coarsest- relaxation numbers)
              KREL=7
              NPRE=2
              NPOST=1
              NCOARS=100
C
C       (Set SOR relaxation parameter OMEGA)
              DO 110 IPDE=1,NPDES
                W(IPDE)=1.ODO
  110         CONTINUE
C
C       (Set restriction category)
C       (Note  in the program, prolongation is bilinear)
              KREST=3
C
C       (Set w-cycle style)
              NCGC=2
C
C       (Set corner type)
              NCOR=1
C
              GOTO 200
C
            ENDIF
C       ---------
C       ELSE
C       ---------
C       KREL  kind of relaxation method used as a smoother
C                 1    - Point Gauss-Seidel
C                 2    - Symmetric Point Gauss-Seidel
C                 3    - Red-Black
C                 4    - x-Line Gauss-Seidel
C                 5    - y-Line Gauss-Seidel
C                 6    - Alternating Line Gauss-Seidel
C                 7    - Alternating Symmetric Line Gauss-Seidel
C                 8    - x-Line Zebra
C                 9    - y-Line Zebra
C                10    - Alternating Line Zebra
            READ(5,*) KREL
C           -------
C       NPRE  max no of pre-relaxation sweeps
            READ(5,*) NPRE
C           -------
C       NPOST  max no of post-relaxation sweeps
            READ(5,*) NPOST
C           -------
C       NCOARS  max no of relaxations on coarsest mesh
            READ(5,*) NCOARS
C           -------
C       W   relaxation parameter OMEGA for each variable in system
            DO 120 IPDE=1,NPDES
              READ(5,*) W(IPDE)
  120       CONTINUE
C           -------
C       KREST   kind of restriction used
C                   1    - Injection
C                   2    - Half-Weighting    (5 Points)
C                   3    - Full-Weighting    (9 Points)
            READ(5,*) KREST
C           -------
C       NCGC    no of coarse grid corrections
C                  1    - V-Cycles
C                  2    - W-Cycles
            READ(5,*) NCGC
C           -------
C       NCOR    corner volume box type (1,2,or 3)
            READ(5,*) NCOR
C           -------
C
C       (*** END OF PARAMETER INPUTS ***)
C
C
  200     CONTINUE
C     ---------------------------------------
C     Set grid sizes and indication pointer
C     ---------------------------------------
            NX(1)=IE(1)-IB(1)+1
            NY(1)=JE(1)-JB(1)+1
            IST(1)=NX(1)*NY(1)+1
C
            DO 210 IGRID=2,MXNOG
              IGM1=IGRID-1
              IST(IGRID)=IST(IGM1)+NX(IGM1)*NY(IGM1)
              NX(IGRID)=2*NX(IGM1)-1
              NY(IGRID)=2*NY(IGM1)-1
  210       CONTINUE
C     ---------------------------
C     Set total maximum size of Q
C     ---------------------------
            MAXTOT=IST(MXNOG)+NX(MXNOG)*NY(MXNOG)
C     ------------------------------------------------------------------
C     Set sub maximum size of auxiliary variable on any grid level
C     ------------------------------------------------------------------
            MAXSUB=NX(1)*NY(1)
            DO 220 IGRID=2,MXNOG
              MAXSUB=MAX0(MAXSUB,NX(IGRID)*NY(IGRID))
  220       CONTINUE
C     -----------------------------------------------
C     Initialize solution and R.H.S. arrays Q and FF
C     -----------------------------------------------
            DO 230 MM=1,MAXTOT
              DO 230 IPDE=1,NPDES
                Q(MM,IPDE)=0.ODO
                FF(MM,IPDE)=0.ODO
  230       CONTINUE
C     ---------------------------
C     Set Red-Black Zebra Switch
C     ---------------------------
            ID=1
            IF ((KREL.EQ.3) OR (KREL.GE.8)) ID=2
C     -----------------------------
C     Set Restriction Coefficients
C     -----------------------------
C     (If injection has been used, i.e. KREST=1)
            R1=1.0DO
```

```
      R2=0.0D0
      R3=0.0D0
C
C     (If half weighting has been used)
      IF (KREST.EQ.2) THEN
        R1=.5D0
        R2=.125D0
        R3=0.0D0
      ENDIF
C
C     (If full weighting has been used)
      IF (KREST.EQ.3) THEN
        R1=0.25D0
        R2=.5D0*R1
        R3=0.5D0*R2
      ENDIF
C
C     ------------------------------------------------
C     Check users choice of Smoother For Compatability
C     ------------------------------------------------
C
C     (Injection or half-weighting is not good for Red-Black
C      Gauss-Seidel method)
      IF (KREST.LE.2) THEN
        IF (KREL.EQ.3) WRITE(6,9000)
        IF (KREL.GE.8) WRITE(6,9000)
      ENDIF
C
 9000 FORMAT(1X,/,'WARNING  INJECTION OR HALF-WEIGHTING IS
     &     NOT SUITABLE FOR THE RED-BLACK METHOD ',/)
C
      RETURN
      END
C
C
C
      SUBROUTINE MG(NOG)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON U(50000,2),NX(10),NY(10),IS(10)
      COMMON /NOPDES/ MXNOG,NPDES
      COMMON /MISC/ TOLRES,MAXITS,KREST,NCGC
      COMMON /RELP/ KREL,NPRE,NPOST,NCOARS
      COMMON /OMEGA/ W(2)
      COMMON /UNILEV/ LEV
      DIMENSION TOL(10)
      INTEGER ICGC(10)
      CHARACTER*70 DESC
C
C     --------------------
C     Initialize parameters
C     --------------------
      RMG=1.0D0
C
C     -------------------------------
C     Fix the present finest boundary
C     -------------------------------
      CALL BCS(NOG)
C
C     ----------------------------
C     Main frame with nested method
C     ----------------------------
      DO 1000 MGIT=1,MAXITS
C
C     ------------------------------------------------------
C     Initialize the tolerance criterion on all the levels
C     ------------------------------------------------------
      DO 10 I=1,NOG
        TOL(I)=TOLRES
   10 CONTINUE
C
C     ----------------------------------------
C     Initialize no. of cycles on all the levels
C     ----------------------------------------
      DO 20 I=1,NOG-1
        ICGC(I)=0
   20 CONTINUE
C
      K=NOG
C
C     -------------------------------
C     Fix present finest level R H S
C     -------------------------------
      CALL DEF(K,0)
C
C     ------------------------------------------------------
C     Initialize the no. of relaxation on the coarsest grid
C     ------------------------------------------------------
      NNCOAS=0
C
C     ---------------------------------------
C     If only one level (coarsest) presents
C     ---------------------------------------
      IF (NOG.EQ.1) GOTO 300
C
C     ----------
C     MAIN CYCLE
C     ----------
C
C     --------------
C     Pre-relaxation
C     --------------
  100 CONTINUE
C
      DO 110 IREL=1,NPRE
        CALL RELAX(K,NOG,EN)
  110 CONTINUE
C
C     ----------------------
C     Restrict from K to K-1
C     ----------------------
  200 K=K-1
C
C     ---------------------------------
C     Fix the R H S  on the coarser level
C     ---------------------------------
      CALL DEF(K,1)
C
      ICGC(K)=ICGC(K)+1
C
C     -------------------------------
C     Fix tolerance on the coarser grid
C     -------------------------------
```

University of Alberta

```
      IF (K LT LEV) THEN
        TOL(K)*DMAX1(10 2D0*EN,TOLRES)
      ELSE
        TOL(K)*4 0D0*EN
      ENDIF
C
      IF (K EQ 1) TOL(K)*TOLRES
      IF (K NE 1) GOTO 100
C
  300 CONTINUE
C
C     ----------------------
C     Coarsest grid solution
C     ----------------------
      DO 310 IREL*1,NCOARS
        CALL RELAX(K,NOG,EN)
        IIREL*IREL
        IF (EN LT TOL(K)) GOTO 320
  310 CONTINUE
C
  320 IF (NOG EQ 1) THEN
        NNCOAS*NNCOAS*IIREL
        GOTO 800
      ENDIF
C
  400 CONTINUE
C
C     ------------------------------------------------------
C     Prolong from K to K+1 (including the interior boundary)
C     ------------------------------------------------------
      CALL PROLIK,II
      K*K+1
C
C     ---------------
C     Post-relaxation
C     ---------------
      DO 500 IREL*1,NPOST
        CALL RELAX(K,NOG,EN)
  500 CONTINUE
C
C     ------------------------------------
C     Terminate iteration if on finest grid
C     ------------------------------------
      IF (K EQ NOG) GOTO 600
C
C     ------------------------------------------------
C     Repeat coarse grid correction if W-cycle required
C     ------------------------------------------------
C
      IF (ICGC(K-1) NE NCGC) GOTO 100
      ICGC(K-1)*0
      GOTO 400
C
  600 CONTINUE
C
      RMGO*RMG
      RMG*EN
      CMG*RMG/RMGO
      IF (NOG EQ 1) THEN
        PSM*CMG**(1 0D0/DFLOAT(NNCOAS))
      ELSE
        PSM*CMG**(1 0D0/DFLOAT(NPRE+NPOST))
      ENDIF
      IF (MGIT NE 1) WRITE(6,8008) CMG,PSM
C
  700 CONTINUE
C
      IF (EN LT TOL(NOG)) THEN
        WRITE(6,8003) MGIT
        DESC*'Final Solution'
        CALL OUTSOL(NOG,DESC)
        GOTO 8000
      ENDIF
C
 1000 CONTINUE
C
      WRITE(6,8004) MAXITS
      STOP
C
 8000 CONTINUE
C
C
 8003 FORMAT(1X,//,' Converged Within Tolerance In ',I4,' MG Itns')
 8004 FORMAT(1X,//,' Failed To Converge In ',I4,' MG Itns')
 8008 FORMAT(1X,/,' MG Convergence Rate =',D16 8,/,
     &            ' Practical Smoothing Rate =',D16 8)
C
      RETURN
      END
C
C
C
C     EXAMPLE - TWO-DIMENSIONAL POISSON EQUATION
C     ******************************************
      SUBROUTINE FNIK,I,J,IO,F,IPDE,JAC,DFDO)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON Q(50000,3),NX(10),NY(10),IST(10)
      DIMENSION F(3),DFDO(2,2,5)
      LOGICAL JAC
C
C     ------------------------------------
C     In L U=F, define L U (on inner points)
C       n                 n
C     ------------------------------------
C
      N=NX(K)
C
      F(IPDE)=Q(IO-N,IPDE)+Q(IO-1,IPDE)+Q(IO+1,IPDE)
     &        +Q(IO+N,IPDE)-4 0D0*Q(IO,IPDE)
C
      IF (JAC) THEN
        DFDO(1,1,1)=-4 0D0
        DFDO(1,1,2)=1 0D0
        DFDO(1,1,3)=1 0D0
        DFDO(1,1,4)=1 0D0
        DFDO(1,1,5)=1 0D0
      ENDIF
C
      RETURN
      END
C
C
```

```
      C
            SUBROUTINE BCS(K)
            IMPLICIT REAL*8 (A-H,O-Z)
            COMMON Q(50000,2),NX(10),NY(10),IST(10)
            COMMON /RANGE/ IB(10),IE(10),JB(10),JE(10)
            COMMON /NOPDES/ MXNOG,NPDES
            COMMON /PARAM/ EPS
      C
      C     ------------------------------------
      C     Define boundary on the whole region
      C     ------------------------------------
      C
            N=NX(K)
            EPS=1.0D0/EPS
            H=1.0D0/DFLOAT(N-1)
      C
            DO 100 J=JB(K),JE(K)
              DO 100 I=IB(K),IE(K)
              IJ=IST(K)+(J-1)*N+I-1
              DO 100 IPDE=1,NPDES
                IF (I.EQ.1) THEN
                  Q(IJ,IPDE)=DEXP(-EPS*(1.0D0+(1.0D0-DFLOAT(J-1)*H)**2))
                  GO TO 90
                ENDIF
                IF (J.EQ.1) THEN
                  Q(IJ,IPDE)=DEXP(-EPS*(1.0D0+(1.0D0-DFLOAT(I-1)*H)**2))
                  GO TO 90
                ENDIF
                IF (I.EQ.N) THEN
                  Q(IJ,IPDE)=DEXP(-EPS*(1.0D0-DFLOAT(J-1)*H)**2)
                  GO TO 90
                ENDIF
                IF (J.EQ.N) THEN
                  Q(IJ,IPDE)=DEXP(-EPS*(1.0D0-DFLOAT(I-1)*H)**2)
                ENDIF
   90         CONTINUE
  100       CONTINUE
      C
            RETURN
            END
      C
      C
      C
            DOUBLE PRECISION FUNCTION FA1(X)
            IMPLICIT REAL*8 (A-H,O-Z)
            COMMON /PARAM/ EPS
      C
            FA1=DEXP(-EPS*(1.0D0-X)**2)
      C
            RETURN
            END
      C
      C
      C
            DOUBLE PRECISION FUNCTION FA2(X)
            IMPLICIT REAL*8 (A-H,O-Z)
            COMMON /PARAM/ EPS
      C
            Y=-EPS*(1.0D0-X)**2
            FA2=Y*DEXP(Y)
      C
            RETURN
            END
      C
      C
      C
            SUBROUTINE FRMS(K,I,J,IJ,MSW)
            IMPLICIT REAL*8 (A-H,O-Z)
            COMMON Q(50000,2),NX(10),NY(10),IST(10)
            COMMON /NOPDES/ MXNOG,NPDES
            COMMON /DEFF/ FF(50000,2)
            COMMON /CORNER/ NCOR
            COMMON /PARAM/ EPS
            EXTERNAL FA1,FA2
      C
      C     ------------------
      C     In L def, define f
      C           n
      C     ------------------
            N=NX(K)
            H=1.0D0/DFLOAT(N-1)
            II=I-1
            JJ=J-1
      C
      C     --------------------------
      C     MSW is the switch pointer
      C     --------------------------
      C
      C     MSW=1  inner points
      C     -------------------
            IF (MSW.EQ.1) THEN
              SNLF=(DFLOAT(II)-0.5D0)*H
              SNRT=(DFLOAT(II)+0.5D0)*H
              SNLO=(DFLOAT(JJ)-0.5D0)*H
              SNUP=(DFLOAT(JJ)+0.5D0)*H
            ENDIF
      C
      C     MSW=2  left interior boundary
      C     -----------------------------
            IF (MSW.EQ.2) THEN
              SNLF=DFLOAT(II-1)*H
              SNRT=(DFLOAT(II)+0.5D0)*H
              SNLO=DFLOAT(JJ-1)*H
              SNUP=DFLOAT(JJ+1)*H
            ENDIF
      C
      C     MSW=3  lower interior boundary
      C     ------------------------------
            IF (MSW.EQ.3) THEN
              SNLF=DFLOAT(II-1)*H
              SNRT=DFLOAT(II+1)*H
              SNLO=DFLOAT(JJ-1)*H
              SNUP=(DFLOAT(JJ)+0.5D0)*H
            ENDIF
      C
      C     MSW=4  lower left corner
      C     ------------------------
            IF (MSW.EQ.4) THEN
              SNLF=DFLOAT(II-1)*H
      C
                IF (NCOR.EQ.1) THEN
                  SNRT=(DFLOAT(II)+0.5D0)*H
```

```
                ELSE
                  BNRT=DFLOAT(II+1)*H
                ENDIF
C
                BNLO=DFLOAT(JJ-1)*H
C
                IF (NCOR EQ 1) THEN
                  BNUP=(DFLOAT(JJ)+0.5D0)*H
                ELSE
                  BNUP=DFLOAT(JJ+1)*H
                ENDIF
C
              ENDIF
C
              RERR=1.0D-3
              AERR=0.0D0
C
              C1=DCADRE(FA1,BNLF,BNRT,AERR,RERR,ERROR,IER)
              C2=DCADRE(FA1,BNLO,BNUP,AERR,RERR,ERROR,IER)
              C3=DCADRE(FA2,BNLF,BNRT,AERR,RERR,ERROR,IER)
              C4=DCADRE(FA2,BNLO,BNUP,AERR,RERR,ERROR,IER)
C
              PP=-4.0D0*EPS
C
              DO 100 IPDE=1,NPDES
                FF(IO,IPDE)=PP*(C1*C4+C2*C3+C1*C2)
  100         CONTINUE
C
              RETURN
              END
C
C
C
              SUBROUTINE DEF(K,NSW)
              IMPLICIT REAL*8 (A-H,O-Z)
              COMMON U(50000,2),NX(10),NY(10),IST(10)
              COMMON /RANGE/ IB(10),IE(10),JB(10),JE(10)
              COMMON /UNILEV/ LEV
              COMMON /WORKK/ WT(50000,2)
              COMMON /RESTP/ R1,R2,R3
              COMMON /DEFF/ FF(50000,2)
              COMMON /CORNER/ NCOR
              COMMON /NOPDES/ MXNDC,NPDES
              DIMENSION F(2),DF00(2,2,5)
C
C             ------------------------
C             Define R.H.S  on grid K
C             ------------------------
C
C             -------------------------
C             NSW is the switch pointer
C             -------------------------
C
              N=NX(K)
C             -------------------------
C             NSW=0  for relaxation only
C             -------------------------
              IF (NSW.EQ.0) THEN
                DO 100 J=JB(K)+1,JE(K)-1
                  JM=J-1
                DO 100 I=IB(K)+1,IE(K)-1
                  IM=I-1
C
                  IO=IST(K)+JM*N+IM
                  CALL FRHS(K,I,J,IO,1)
C
  100           CONTINUE
                GO TO 999
              ENDIF
C             ------------------------------------------------------
C             NSW=1  Calculate adjusted R.H.S  on the coraser grid,
C             residual has been considered (on partial points)
C             ------------------------------------------------------
C
              KF=K+1
              NF=NX(KF)
              IBC=(IB(KF)+1)/2
              IEC=(IE(KF)+1)/2
              JBC=(JB(KF)+1)/2
              JEC=(JE(KF)+1)/2
C
              Q1=R1
              Q2=R2
              Q3=R2
              Q4=R3
C             ---------
C             K    K+1
C             R    U
C              K+1
C             ---------
              DO 200 J=JBC,JEC
                JM=J-1
              DO 200 I=IBC,IEC
                IM=I-1
C
                IO=IST(K)+JM*N+IM
                JO=IST(KF)+2*(JM*NF+IM)
                JOP=JO+NF
                JOM=JO-NF
C
                DO 200 IPDE=1,NPDES
                  IF ((I.GT.IBC).AND.(I.LT.IEC).AND.(J.GT.JBC).AND.
     &                (J.LT.JEC)) THEN
C                   (full weighting on the inner points)
                    U(IO,IPDE)=Q1*U(JO,IPDE)+Q2*(U(JO+1,IPDE)+U(JO-1,IPDE))
     &                         +Q3*(U(JOP,IPDE)+U(JOM,IPDE))+Q4*(U(JOP+1,IPDE)
     &                         +U(JOP-1,IPDE)+U(JOM+1,IPDE)+U(JOM-1,IPDE))
                  ELSE
C                   (injection on the interior boundary)
                    U(IO,IPDE)=U(JO,IPDE)
                  ENDIF
  200           CONTINUE
C
C             calculate the residual in the inner points of the local
C             region
C             -------------------------------------------------------
              CALL RESFF(KF)
C
C             calculate the R.H.S  on the coarser grid
```

University of Alberta

```
      C     .....................................
            DO 300 J=JB(K)+1,JE(K)-1
              JM=J-1
            DO 300 I=IB(K)+1,IE(K)-1
              IM=I-1
      C
              IO=IST(K)+JM*N+IM
              JO=IST(KF)+2*(JM*NF+IM)
              JOP=JO+NF
              JOM=JO-NF
      C
              IF ((I.LT.IBC).OR.(I.GT.IEC).OR.
     &            (J.LT.JBC).OR.(J.GT.JEC)) THEN
      C       .....................................
      C         out of local region  G(K,N)/G(K+1,N)
      C       .....................................
                CALL FRMS(K,I,J,IO,1)
                GO TO 290
              ENDIF
      C
              IF ((I.GT.IBC).AND.(I.LT.IEC).AND.(J.GT.JBC)
     &          .AND.(J.LT.JEC)) THEN
      C       .....................................
      C         inner points in the local region  G(K+1,N)
      C       .....................................
                DO 210 IPDE=1,NPDES
      C         ---------------------------------------------------------
      C                                        K    K+1 K+1   K+1
      C           Calculate residual  I  (F  - L   U   ) (full weighting)
      C                                       K+1
      C         ---------------------------------------------------------
                FF(IO,IPDE)=.01*WT(JO,IPDE)+.02*(WT(JO+1,IPDE)
     &              +WT(JO-1,IPDE))+.03*(WT(JOP,IPDE)+WT(JOM,IPDE))
     &              +.04*(WT(JOP+1,IPDE)+WT(JOP-1,IPDE)
     &              +WT(JOM+1,IPDE)+WT(JOM-1,IPDE))
      C         .....................................
      C           Impose residual onto the right hand side
      C         .....................................
                CALL FNIK(I,J,IO,F,IPDE, FALSE ,DFDO)
                FF(IO,IPDE)=FF(IO,IPDE)+F(IPDE)
        210     CONTINUE
                GO TO 280
              ENDIF
      C
              IF ((I.EQ.IBC).AND.(J.GT.JBC).AND.(J.LT.JEC)) THEN
      C       .....................................
      C         on left boundary of the local region
      C       .....................................
                CALL FRMS(KF,2*I-1,2*J-1,JO,2)
                DO 220 IPDE=1,NPDES
                  FF(IO,IPDE)=FF(JO,IPDE)-(Q(IO-1,IPDE)-2.0D0*Q(JO,IPDE)
     &              +Q(JO+1,IPDE))*.75D0+(Q(JOM-NF,IPDE)
     &              +Q(JOP+NF,IPDE)-2.0D0*Q(JO,IPDE))
     &              -.5D0*(Q(JOM+1,IPDE)+Q(JOP+1,IPDE)
     &              -Q(JOM,IPDE)-Q(JOP,IPDE))
                  CALL FNIK(I,J,IO,F,IPDE, FALSE ,DFDO)
                  FF(IO,IPDE)=FF(IO,IPDE)+F(IPDE)
        220       CONTINUE
                  GO TO 280
                ENDIF
      C
              IF ((J.EQ.JBC).AND.(I.GT.IBC).AND.(I.LT.IEC)) THEN
      C       .....................................
      C         on lower boundary of the local region
      C       .....................................
                CALL FRMS(KF,2*I-1,2*J-1,JO,3)
                DO 230 IPDE=1,NPDES
                  FF(IO,IPDE)=FF(JO,IPDE)-(Q(IO-N,IPDE)-2.0D0
     &              +Q(JO,IPDE)+Q(JOP,IPDE))*.75D0+(Q(JO+2,IPDE)
     &              +Q(JO-2,IPDE)-2.0D0*Q(JO,IPDE))
     &              -.5D0*(Q(JOP+1,IPDE)+Q(JOP-1,IPDE)
     &              -Q(JO-1,IPDE)-Q(JO+1,IPDE))
                  CALL FNIK(I,J,IO,F,IPDE, FALSE ,DFDO)
                  FF(IO,IPDE)=FF(IO,IPDE)+F(IPDE)
        230       CONTINUE
                  GO TO 280
                ENDIF
      C
              IF ((I.EQ.IBC).AND.(J.EQ.JBC)) THEN
      C       .....................................
      C         on lower left corner of the local region
      C       .....................................
                CALL FRMS(KF,2*I-1,2*J-1,JO,4)
                DO 240 IPDE=1,NPDES
                  IF (NCOR.EQ.1) THEN
                    FF(IO,IPDE)=FF(JO,IPDE)-.75D0*(Q(IO-1,IPDE)
     &                +Q(IO-N,IPDE)-2.0D0*Q(JO,IPDE))
     &                -1.5D0*(Q(JO+1,IPDE)-2.0D0*Q(JO,IPDE)
     &                +Q(JOP,IPDE))
                    GO TO 235
                  ENDIF
                  IF (NCOR.EQ.2) THEN
                    FF(IO,IPDE)=FF(JO,IPDE)-(Q(IO-1,IPDE)+Q(IO-N,IPDE)
     &                -2.0D0*Q(JO,IPDE))-.75D0*(Q(JO+2,IPDE)
     &                +Q(JOP+NF,IPDE)-2.0D0*Q(JO,IPDE))
     &                -.25D0*(Q(JOP+2,IPDE)+Q(JOP+NF+1,IPDE)
     &                -Q(JOP,IPDE)-Q(JO+1,IPDE))
                    GO TO 235
                  ENDIF
                  IF (NCOR.EQ.3) THEN
                    FF(IO,IPDE)=FF(JO,IPDE)-(Q(IO-1,IPDE)+Q(IO-N,IPDE)
     &                -4.0D0*Q(JO,IPDE))-(Q(JO+2,IPDE)+Q(JOP+NF,IPDE))
                  ENDIF
        235       CONTINUE
                  CALL FNIK(I,J,IO,F,IPDE, FALSE ,DFDO)
                  FF(IO,IPDE)=FF(IO,IPDE)+F(IPDE)
        240       CONTINUE
              ENDIF
      C
        290   CONTINUE
      C
        300   CONTINUE
      C
        888   CONTINUE
      C
              RETURN
              END
      C
      C
      C
              SUBROUTINE RESFF(K)
              IMPLICIT REAL*8 (A-H,O-Z)
```

```
Listing of LMRA FOR at 03 15 23 on APR 12, 1980 for CC1@KCHN on UALTAMTS

      855            COMMON Q(50000,2),NX(10),NY(10),IST(10)
      856            COMMON /RANGE/ IB(10),IE(10),JB(10),JE(10)
      857            COMMON /NOPDES/ MXNOG,NPDES
      858            COMMON /WORKK/ WT(50000,2)
      859            COMMON /DEFF/ FF(50000,2)
      860            DIMENSION F(2),DFDQ(2,2,5)
      861      C
      862            DO 10 J=JB(K)+1,JE(K)-1
      863              JM=J-1
      864            DO 10 I=IB(K)+1,IE(K)-1
      865              IM=I-1
      866      C
      867              IQ=IST(K)+JM*NX(K)+IM
      868      C
      869              DO 10 IPDE=1,NPDES
      870      C                                                          N+1  N+1 N+1
      871      C
      872      C               In inner points of finer grid, calculate F   L   U
      873      C               ------------------------------------------------------
      874              CALL FN(K,I,J,IQ,F,IPDE, FALSE ,DFDQ)
      875      C
      876              WT(IQ,IPDE)=FF(IQ,IPDE)-F(IPDE)
      877       10   CONTINUE
      878      C
      879            RETURN
      880            END
      881      C
      882      C
      883      C
      884            SUBROUTINE RELAX(K,NOG,EN)
      885            IMPLICIT REAL*8 (A-H,O-Z)
      886            COMMON Q(50000,2),NX(10),NY(10),IST(10)
      887            COMMON /RANGE/ IB(10),IE(10),JB(10),JE(10)
      888            COMMON /RELP/ KREL,NPRE,NPOST,NCOARS
      889            COMMON /RB/ ID
      890      C
      891            EN=0.0D0
      892      C
      893            IF(KREL.GT.3) GOTO 30
      894      C
      895      C     Point Relaxation Schemes
      896      C     ------------------------
      897            DO 10 J=JB(K)+1,JE(K)-1
      898            JM=J-1
      899            DO 10 I=IB(K)+1,IE(K)-1
      900            IM=I-1
      901            IQ=IST(K)+JM*NX(K)+IM
      902       10 IF(MOD(I+J,ID).EQ.0) CALL POINTS(K,I,J,IQ,EN)
      903      C
      904            IF(KREL.EQ.1) GOTO 100
      905            IF(KREL.EQ.2) THEN
      906            EN=0.0D0
      907            ENDIF
      908      C
      909            DO 20 J=JE(K)-1,JB(K)+1,-1
      910            JM=J-1
      911            DO 20 I=IE(K)-1,IB(K)+1,-1
      912            IM=I-1
      913            IQ=IST(K)+JM*NX(K)+IM
      914       20 IF(MOD(I+J+1,ID).EQ.0) CALL POINTS(K,I,J,IQ,EN)
      915      C
      916            GOTO 100
      917      C
      918       30 CONTINUE
      919      C
      920      C     Line Relaxation Schemes
      921      C     -----------------------
      922      C
      923            IF((KREL.EQ.5).OR.(KREL.EQ.9)) GOTO 60
      924      C
      925            DO 40 J=JB(K)+1,JE(K)-1
      926       40 IF(MOD(J,ID).EQ.0) CALL LINES(K,I,J,1,EN)
      927      C
      928            IF(KREL.EQ.4) GOTO 100
      929            IF(KREL.EQ.8) GOTO 60
      930            IF(KREL.EQ.7) THEN
      931            EN=0.0D0
      932            ENDIF
      933      C
      934            DO 50 J=JE(K)-1,JB(K)+1,-1
      935       50 IF(MOD(J+1,ID).EQ.0) CALL LINES(K,I,J,1,EN)
      936      C
      937            IF(KREL.EQ.8) GOTO 100
      938      C
      939       60 IF((KREL.EQ.6).OR.(KREL.EQ.10).OR.(KREL.EQ.7)) THEN
      940            EN=0.0D0
      941            ENDIF
      942      C
      943            DO 70 I=IB(K)+1,IE(K)-1
      944       70 IF(MOD(I,ID).EQ.0) CALL LINES(K,I,J,2,EN)
      945      C
      946            IF((KREL.EQ.5).OR.(KREL.EQ.9)) GOTO 100
      947            IF(KREL.EQ.7) THEN
      948            EN=0.0D0
      949            ENDIF
      950      C
      951            DO 80 I=IE(K)-1,IB(K)+1,-1
      952       80 IF(MOD(I+1,ID).EQ.0) CALL LINES(K,I,J,2,EN)
      953      C
      954       100 CONTINUE
      955      C
      956            WRITE(6,9000) K,EN
      957      9000 FORMAT(1X,'GRID',I4,3X,'RESIDUAL NORM=',D18.8)
      958      C
      959            RETURN
      960            END
      961      C
      962      C
      963      C
      964            SUBROUTINE POINTS(K,I,J,IQ,EN)
      965            IMPLICIT REAL*8 (A-H,O-Z)
      966            COMMON Q(50000,2),NX(10),NY(10),IST(10)
      967            COMMON /DEFF/ FF(50000,2)
      968            COMMON /NOPDES/ MXNOG,NPDES
      969            COMMON /OMEGA/ W(2)
      970            DIMENSION F(2),R(2),S(2),DFDQ(2,2,5)
      971      C
      972            DO 10 IPDE=1,NPDES
      973              CALL FN(K,I,J,IQ,F,IPDE, TRUE ,DFDQ)
      974              R(IPDE)=FF(IQ,IPDE)-F(IPDE)
      975              EN=DMAX1(EN,DABS(R(IPDE)))
      976      C
      977       10 CONTINUE
      978      C
```

```
      CALL MINVV(NPDES,DFDO,R,R)
C
      DO 20 IPDE=1,NPDES
        Q(IO,IPDE)=Q(IO,IPDE)-W(IPDE)*R(IPDE)
   20 CONTINUE
C
      RETURN
      END
C
C
C
      SUBROUTINE MINVV(NPDES,A,X,B)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(2,2),X(2),B(2)
C
      IF(NPDES.EQ.1) THEN
        X(1)=B(1)/A(1,1)
        RETURN
      ENDIF
C
      IF(NPDES.EQ.2) THEN
        DET=A(1,1)*A(2,2)-A(1,2)*A(2,1)
        RDET=1.0D0/DET
        X(1)=RDET*(A(2,2)*B(1)-A(1,2)*B(2))
        X(2)=RDET*(A(1,1)*B(2)-A(2,1)*B(1))
        RETURN
      ENDIF
C
      RETURN
      END
C
C
C
      SUBROUTINE LINES(K,I,J,IEY,EN)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON Q(50000,2),NX(10),NY(10),IST(10)
      COMMON /RANGE/ IB(10),IE(10),JB(10),JE(10)
      COMMON /NOPDES/ MXNOC,NPDES
      COMMON /OMEGA/ W(2)
      COMMON /DEFF/ FF(50000,2)
      COMMON /TRISOL/ A0(2100,2,2),A1(2100,2,2),A2(2100,2,2),
     &                X(2100,2),R(2100,2)
      DIMENSION F(2),DFDO(2,2,6)
C
C     ---------------------------
C     IEY  =1,x-dir'n, =2,y-dir'n
C     ---------------------------
C
      IF (IEY.EQ.1) THEN
C
        JM=J-1
        DO 100 I=IB(K)+1,IE(K)-1
          IM=I-1
          IO=IST(K)+JM*NX(K)+IM
          DO 100 IPDE=1,NPDES
            CALL FN(K,I,J,IO,F,IPDE, TRUE ,DFDO)
            R(I,IPDE)=F(IPDE)-FF(IO,IPDE)
            EN=DMAX1(EN,DABS(R(I,IPDE)))
            DO 100 JPDE=1,NPDES
              A0(I,IPDE,JPDE)=DFDO(IPDE,JPDE,1)
              A1(I,IPDE,JPDE)=DFDO(IPDE,JPDE,2)
              A2(I,IPDE,JPDE)=DFDO(IPDE,JPDE,3)
  100   CONTINUE
C
        CALL TRIDIG(NPDES,K,1)
C
        DO 200 I=IB(K)+1,IE(K)-1
          IM=I-1
          IO=IST(K)+JM*NX(K)+IM
          DO 200 IPDE=1,NPDES
            Q(IO,IPDE)=Q(IO,IPDE)-W(IPDE)*X(I,IPDE)
  200   CONTINUE
C
      ELSE
C
        IM=I-1
        DO 300 J=JB(K)+1,JE(K)-1
          JM=J-1
          IO=IST(K)+JM*NX(K)+IM
          DO 300 IPDE=1,NPDES
            CALL FN(K,I,J,IO,F,IPDE, TRUE ,DFDO)
            R(J,IPDE)=F(IPDE)-FF(IO,IPDE)
            EN=DMAX1(EN,DABS(R(J,IPDE)))
            DO 300 JPDE=1,NPDES
              A0(J,IPDE,JPDE)=DFDO(IPDE,JPDE,1)
              A1(J,IPDE,JPDE)=DFDO(IPDE,JPDE,4)
              A2(J,IPDE,JPDE)=DFDO(IPDE,JPDE,6)
  300   CONTINUE
C
        CALL TRIDIG(NPDES,K,2)
C
        DO 400 J=JB(K)+1,JE(K)-1
          JM=J-1
          IO=IST(K)+JM*NX(K)+IM
          DO 400 IPDE=1,NPDES
            Q(IO,IPDE)=Q(IO,IPDE)-W(IPDE)*X(J,IPDE)
  400   CONTINUE
C
      ENDIF
C
      RETURN
      END
C
C
C
      SUBROUTINE TRIDIG(NPDES,K,IEY)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /RANGE/ IB(10),IE(10),JB(10),JE(10)
      COMMON /TRISOL/ A0(2100,2,2),A1(2100,2,2),A2(2100,2,2),
     &                X(2100,2),R(2100,2)
      DIMENSION D(2,2)
C
C     ------------------------------------------------
C     Pre-treatement  IEY=1  x-dir'n,  IEY=2  y-dir'n
C     ------------------------------------------------
      IF (IEY.EQ.1) THEN
        IJB=IB(K)+1
        IJE=IE(K)-1
      ELSE
        IJB=JB(K)+1
        IJE=JE(K)-1
      ENDIF
```

University of Alberta

```
      C
      C     LU Decomposition
      C     ----------------
            CALL MINVA(NPDES,IJB)
            DO 30 I=IJB+1,IJE
              DO 10 IPDE=1,NPDES
              DO 10 JPDE=1,NPDES
                D(IPDE,JPDE)=0.000
              DO 10 KPDE=1,NPDES
                D(IPDE,JPDE)=D(IPDE,JPDE)
     &                      +A2(I,IPDE,KPDE)*A0(I-1,KPDE,JPDE)
     10     CONTINUE
              DO 20 IPDE=1,NPDES
              DO 20 JPDE=1,NPDES
                A2(I,IPDE,JPDE)=D(IPDE,JPDE)
              DO 20 KPDE=1,NPDES
                A0(I,IPDE,JPDE)=A0(I,IPDE,JPDE)
     &                         -D(IPDE,KPDE)*A1(I-1,KPDE,JPDE)
     20     CONTINUE
            CALL MINVA(NPDES,I)
     30   CONTINUE
      C
      C     Forward Substitution
      C     --------------------
            DO 40 I=IJB+1,IJE
            DO 40 IPDE=1,NPDES
            DO 40 KPDE=1,NPDES
              R(I,IPDE)=R(I,IPDE)-A2(I,IPDE,KPDE)*R(I-1,KPDE)
     40   CONTINUE
      C
      C     Back Substitution
      C     -----------------
            DO 50 IPDE=1,NPDES
              X(IJE,IPDE)=0.000
            DO 50 KPDE=1,NPDES
              X(IJE,IPDE)=X(IJE,IPDE)+A0(IJE,IPDE,KPDE)*R(IJE,KPDE)
     50   CONTINUE
      C
            DO 70 I=IJE-1,IJB,-1
              DO 60 IPDE=1,NPDES
              DO 60 KPDE=1,NPDES
                R(I,IPDE)=R(I,IPDE)-A1(I,IPDE,KPDE)*X(I+1,KPDE)
     60       CONTINUE
            DO 70 IPDE=1,NPDES
              X(I,IPDE)=0.000
            DO 70 KPDE=1,NPDES
              X(I,IPDE)=X(I,IPDE)+A0(I,IPDE,KPDE)*R(I,KPDE)
     70   CONTINUE
      C
            RETURN
            END
      C
      C
      C
            SUBROUTINE MINVA(NPDES,I)
            IMPLICIT REAL*8 (A-H,O-Z)
            COMMON /TRISOL/ A0(2100,2,2),A1(2100,2,2),A2(2100,2,2),
     &                      X(2100,2),R(2100,2)
      C
            IF(NPDES.EQ.1) THEN
              A0(I,1,1)=1.000/A0(I,1,1)
            ENDIF
      C
            IF(NPDES.EQ.2) THEN
              DET=A0(I,1,1)*A0(I,2,2)-A0(I,1,2)*A0(I,2,1)
              RDET=1.000/DET
              DUM=A0(I,1,1)
              A0(I,1,1)=RDET*A0(I,2,2)
              A0(I,2,2)=RDET*DUM
              A0(I,1,2)=-RDET*A0(I,1,2)
              A0(I,2,1)=-RDET*A0(I,2,1)
              RETURN
            ENDIF
      C
            RETURN
            END
      C
      C
      C
            SUBROUTINE PROL(K,INIT)
            IMPLICIT REAL*8 (A-H,O-Z)
            COMMON U(50000,2),NX(10),NY(10),IST(10)
            COMMON /DEPP/ PP(50000,2)
            COMMON /RANGE/ IB(10),IE(10),JB(10),JE(10)
            COMMON /RESTR/ R1,R2,R3
            COMMON /NOPDES/ MXNOG,NPDES
            DIMENSION WW(2100,2)
      C
            KF=K+1
            NX=NX(K)
            NF=NX(KF)
      C
            IBC=(IB(KF)+1)/2
            IEC=(IE(KF)+1)/2
            JBC=(JB(KF)+1)/2
            JEC=(JE(KF)+1)/2
      C
            Q1=R1
            Q2=R2
            Q3=R2
            Q4=R3
      C     ------------------------
      C     Array WW is a dummy array
      C     ------------------------
            DO 20 J=JBC,JEC
              JM=J-1
            DO 20 I=IBC,IEC
              IM=I-1
              I0=IST(K)+JM*NX+IM
              DO 10 IPDE=1,NPDES
                WW(I0,IPDE)=U(I0,IPDE)
     10       CONTINUE
     20     CONTINUE
      C     ------------------------------------------
      C     Adjustment from coarse grid to finer grid
      C     ------------------------------------------
            IF (INIT.EQ.1) THEN
              DO 100 J=JBC,JEC
                JM=J-1
              DO 100 I=IBC,IEC
```

University of Alberta

```
Listing of LMRA FORTRAN 03 15 23 on APR 12, 1990 for CC:[0,KCHN] on UALTAMTS

 1221                  IM=I-1
 1222                  IO=1ST(K)+JM*N+IM
 1223                  JO=1ST(KF)+2*JM*NF+2*IM
 1224                  JOP=JO+NF
 1225                  JOM=JO-NF
 1226                  IF ((I EQ IBC) OR (I EQ IEC) OR
 1227                 &       (J EQ JBC) OR (J EQ JEC)) THEN
 1228           C      ------------------------------
 1229           C      On the inner boundary points
 1230           C      ------------------------------
 1231                  DO 80 IPDE=1,NPDES
 1232                    WW(IO,IPDE)=WW(IO,IPDE)-Q(JO,IPDE)
 1233            80    CONTINUE
 1234           C
 1235                  ELSE
 1236           C
 1237           C      ------------------------------------------------
 1238           C      On the inner points  ( u(k)-I(k+1)*u(k+1) )
 1239           C      ------------------------------------------------
 1240                  DO 90 IPDE=1,NPDES
 1241                    WW(IO,IPDE)=WW(IO,IPDE)-Q1*Q(JO,IPDE)
 1242                 &       -Q2*(Q(JO+1,IPDE)+Q(JO-1,IPDE))
 1243                 &       -Q3*(Q(JOP,IPDE)+Q(JOM,IPDE))-Q4*(Q(JOP+1,IPDE)+
 1244                 &       Q(JOP-1,IPDE)+Q(JOM+1,IPDE)+Q(JOM-1,IPDE))
 1245            90    CONTINUE
 1246                  ENDIF
 1247           C
 1248            100   CONTINUE
 1249           C
 1250                ENDIF
 1251           C
 1252           C      ------------------------------------------------------
 1253           C      If INIT#0, interpolating solution from old finest grid
 1254           C      to new finest grid only
 1255           C      ------------------------------------------------------
 1256                DO 200 J=JBC,JEC
 1257                  JM=J-1
 1258                DO 200 I=IBC,IEC
 1259                  IM=I-1
 1260                  IO=1ST(K)+JM*N+IM
 1261                  JO=1ST(KF)+2*JM*NF+2*IM
 1262           C
 1263                DO 200 IPDE=1,NPDES
 1264                  Q(JO,IPDE)=DFLOAT(INIT)*Q(JO,IPDE)+WW(IO,IPDE)
 1265                  IF (I NE IEC) THEN
 1266                    S1= SDO*(WW(IO,IPDE)+WW(IO+1,IPDE))
 1267                    Q(JO+1,IPDE)=DFLOAT(INIT)*Q(JO+1,IPDE)+S1
 1268                  ENDIF
 1269                  IF (J NE JEC) THEN
 1270                    S2= SDO*(WW(IO,IPDE)+WW(IO+N,IPDE))
 1271                    Q(JO+NF,IPDE)=DFLOAT(INIT)*Q(JO+NF,IPDE)+S2
 1272                  ENDIF
 1273                  IF ((I EQ IEC) OR (J EQ JEC)) GOTO 190
 1274                  S3= SDO*(WW(IO+N,IPDE)+WW(IO+N+1,IPDE))
 1275                  Q(JO+NF+1,IPDE)=DFLOAT(INIT)*Q(JO+NF+1,IPDE)+ SDO*(S1+S3)
 1276            190   CONTINUE
 1277            200 CONTINUE
 1278           C
 1279                RETURN
 1280                END
 1281           C
 1282           C
 1283           C
 1284                SUBROUTINE OUTSOL(K,DESC)
 1285                IMPLICIT REAL*8 (A-H,O-Z)
 1286                COMMON Q(50000,2),WX(10),WY(10),IST(10)
 1287                COMMON /DEFF/ FF(50000,2)
 1288                COMMON /RANGE/ IB(10),IE(10),JB(10),JE(10)
 1289                COMMON /NOPDES/ MXMOG,NPDES
 1290                CHARACTER*70 DESC
 1291           C
 1292                DO 10 IPDE=1,NPDES
 1293                  WRITE(6,9000) IPDE,K,DESC
 1294                  WRITE(6,9001)
 1295                DO 10 J=JB(K),JE(K)
 1296                  WRITE(6,9002)
 1297                DO 10 I=IB(K),IE(K)
 1298                  IO=1ST(K)+(J-1)*NX(K)+I-1
 1299                  WRITE(6,9003) I,J,IO,Q(IO,IPDE),FF(IO,IPDE)
 1300              10 CONTINUE
 1301           C
 1302            9000 FORMAT(1X,/,' VARIABLE',I4,' ON GRID',I4,/,1X,A70)
 1303            9001 FORMAT(1X,/,5X,'I',5X,'J',8X,'IO',8X,'Q(IO)',15X,'FF',/,64('-'))
 1304            9002 FORMAT(1X)
 1305            9003 FORMAT(2X,I4,2X,I4,2X,I5,2X,D15.8,4X,D15.8)
 1306           C
 1307                RETURN
 1308                END
 1309           C
 1310           C
 1311           C
 1312                SUBROUTINE POSTIN
 1313                IMPLICIT REAL*8 (A-H,O-Z)
 1314                COMMON Q(50000,2),WX(10),WY(10),IST(10)
 1315                COMMON /RANGE/ IB(10),IE(10),JB(10),JE(10)
 1316                COMMON /NOPDES/ MXMOG,NPDES
 1317                COMMON /MISC/ TOLRES,MAXITS,KREST,NCGC
 1318                COMMON /PARAM/ EPS
 1319                DIMENSION EXACT(2100,2),ERR(2100,2)
 1320           C
 1321                K=MXMOG
 1322                N=NX(K)
 1323                H=1.0D0/DFLOAT(N-1)
 1324                XER=TOLRES
 1325           C
 1326                WRITE(6,800) K
 1327                WRITE(6,810)
 1328           C
 1329                DO 100 J=JB(K),JE(K)
 1330                  JM=J-1
 1331                DO 100 I=IB(K),IE(K)
 1332                  IM=I-1
 1333           C
 1334                  IO=1ST(K)+JM*N+IM
 1335                  KO=JM*N+IM
 1336           C
 1337                  DO 10 IPDE=1,NPDES
 1338                    XX=DFLOAT(IM)*H
 1339                    YY=DFLOAT(JM)*H
 1340                    EXACT(KO,IPDE)=DEXP(-EPS*((1.0D0-XX)**2+(1.0D0-YY)**2))
 1341                    ERR(KO,IPDE)=EXACT(KO,IPDE)-Q(IO,IPDE)
 1342                    XER=DMAX1(XER,DABS(ERR(KO,IPDE)))
```

University of Alberta

```
      C
                  WRITE(6,920) I,J,EXACT(KO,IPDE),Q(IO,IPDE),ERR(KO,IPDE)
   10     CONTINUE
  100   CONTINUE
      C
            WRITE(6,930) K,XER
      C
  900   FORMAT(1X,/,'RESULTS ON LEVEL ',I2,' IS ')
  910   FORMAT(1X,/,5X,'I',5X,'J',14X,'EXACT',12X,'Q(I,J)',13X,'ERROR')
  920   FORMAT(1X,/,2(2X,I4),3X,3(D16.8,2X))
  930   FORMAT(1X,/,'THE MAXIMUM ERROR ON LEVEL ',I2,' IS ',D16.8)
      C
            RETURN
            END
```

University of Alberta